Cross Point Data Collection

Obtaining counting data through the CrossCONNECT Access Point

Contents

1 Introduction

Purpose

The purpose of this document is to describe how counting data, collected by Cross Point devices, can be retrieved through the CrossCONNECT Access Point. Developers that want to incorporate the retrievable data into their own systems should find all required information in this document.

Audience

This document is meant for software developers that want to incorporate Cross Point counting data in their own solutions.

1.1 Terms and abbreviations

Terms	Abbreviation
XSD	XML Schema Definition
XML	Extendable Markup Language
W3C	World Wide Web Consortium (http://www.w3.org)
DOM	Document Object Model
SAX	Simple API for XML
API	Application Programming Interface
CSV	Comma Separated Values
FTP	File Transfer Protocol
SDK	Software Development Kit
DTD	Document Type Definition
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation

2 Integration

2.1 About integration

Cross Point gives integrators the possibility to integrate by exchanging data between the Cross Point products or solutions and third party applications. This document will describe the following integration possibilities based on:

1. e-mail or FTP based reports that deliver XML data format
2. direct commands that deliver JSON data format

2.2 Integration requirements

2.2.1 Collecting information

To collect data from the Cross Point platform, a CrossCONNECT Access Point is needed. This CrossCONNECT Access Point collects data from the connected devices and gives users the possibility to obtain all this information and to perform (remote) service on the connected devices.

The information from the devices can be exported in the Cross Point XML output format, so it can be used in other management information systems.

The following items are required to collect data from Cross Point systems.

- CrossCONNECT Access Point
- Device Explorer (software for configuration of the CrossCONNECT Access Point)
- CrossCONNECT enabled systems like: NEXUS and MAXUS devices (such as receivers, transmitters, deactivators, detachers)

2.2.2 Sending information

The Cross Point platform is capable of collecting data from third party applications so this data can be used to enrich the information which is presented by applications like Cross Point Analytics. Examples of information which can be send are visitor counting data or Point-of-Sale data.

To send data to the Cross Point platform a valid API key is needed. This token is used to validate the sender of the data. This key will be supplied be Cross Point or a Cross Point partner.

3 Report based data collection

The Cross Point XML data format can be used by third party applications to collect data from the systems or to send data to Cross Point. The Cross Point XML data can contain the following items:

- Store information
- Device information
- Visitor countings
- Alarms
- Deactivations
- Detachments
- Point-of-Sale information

First a short introduction of the XML language will be given. Then the XML format will be explained and the way to validate the Cross Point XML format. After this an explanation about the way to retrieve the XML from the CrossCONNECT Access Point will be given.

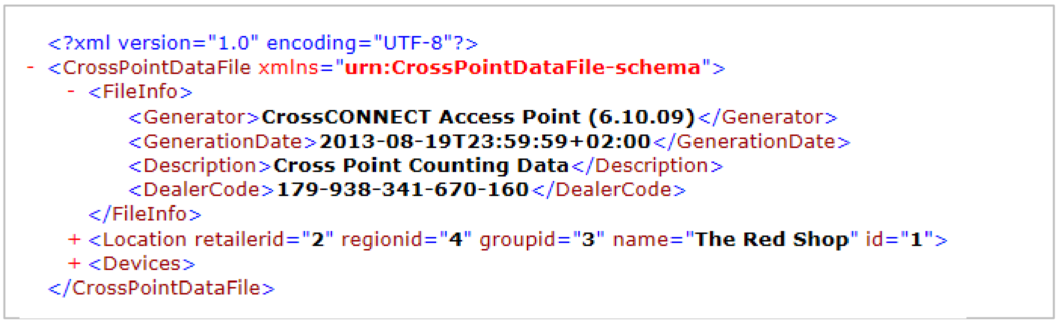
3.1 About XML

XML stands for Extensible Markup Language, and is a general-purpose specification for creating custom markup languages. It is called an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet, and it is used both to encode documents and to serialize data.

XML is recommended by the World Wide Web Consortium. It is a fee-free open standard. The W3C recommendation specifies both the lexical grammar and the requirements for parsing.

In any meaningful application, additional markup is used to structure the contents of the XML document. The text enclosed by the root tags may contain an arbitrary number of XML elements.

In the following example the root tag is **CrossPointDataFile**:



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  - <FileInfo>
        <Generator>CrossCONNECT Access Point (6.10.09)</Generator>
        <GenerationDate>2013-08-19T23:59:59+02:00</GenerationDate>
        <Description>Cross Point Counting Data</Description>
        <DealerCode>179-938-341-670-160</DealerCode>
    </FileInfo>
  + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
  + <Devices>
  </CrossPointDataFile>
```

Figure 1: Root tag example

XML provides special syntax for representing an element with empty content. Instead of writing a start-tag followed immediately by an end-tag, a document may contain an empty-element tag. An empty-element tag resembles a start-tag but contains a slash just before the closing angle bracket.

3.2 Well-formed documents

In XML, a well-formed document must conform to the following rules:

- Non-empty elements are delimited by both a start-tag and an end-tag.
- Empty elements may be marked with an empty-element (self-closing) tag, such as `<IAmEmpty />`. This is equal to `<IAmEmpty></IAmEmpty>`
- All attribute values are quoted with either single (') or double (") quotes. Single quotes close a single quote and double quotes close a double quote.
- Tags may be nested but must not overlap. Each non-root element must be completely contained in another element.
- The document complies with its declared character encoding. The encoding may be declared or implied externally, such as in "Content-Type" headers when a document is transported via HTTP, or internally, using explicit markup at the very beginning of the document. When no such declaration exists, a Unicode encoding is assumed, as defined by a Unicode Byte Order Mark before the document's first character. If the mark does not exist, UTF-8 encoding is assumed.

Element names are case-sensitive. For example, the following is a well-formed matching pair:

```
<Step> ... </Step>
```

whereas this is not:

```
<Step> ... </step>
```

3.3 Automatic verification

It is relatively simple to verify that a document is well-formed or validated XML, because the rules of well-formedness and validation of XML are designed for portability of tools. The idea is that any tool designed to work with XML files will be able to work with XML files written in any XML language (or XML application). One example of using an independent tool follows:

- load it into an XML-capable browser, such as Firefox or Internet Explorer
- use a tool like xmlwf (usually bundled with expat)
- parse the document in your own application

3.4 Processing XML

Three traditional techniques for processing XML files are:

- **Using a programming language and the SAX API**

  SAX is an event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

- **Using a programming language and the DOM API**

  The DOM API is an API that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

- **Using a transformation engine and a filter**

More recent and emerging techniques for processing XML files are:

- **Pull Parsing**

  The pull parsing technique is different from tree-based APIs like the Document Object Model (DOM) and event stream APIs like the Simple API for XML (SAX) in that the application controls the parsing of documents. The application requests, or pulls, the document information one piece at a time instead of having document information pushed to the application.

- **Data binding**

  XML data binding refers to the process of representing the information in an XML document as an object in computer memory. This allows applications to access the data in the XML from the object rather than using the DOM to retrieve the data from a direct representation of the XML itself.

3.5 XML Document information

Cross Point delivers its data in the previously described XML format. This format is flexible, extensible and accepted in many different business environments. It is also a common used format over the internet and more human readable than formats such as CSV. The XML format can also be validated against a DTD (Document Type Description) such as a DTD document or an XML schema, which is not possible with a plain text format like CSV.

The XML document can be generated by the CrossCONNECT Access Point. This device is mainly situated in a retail environment, and collects data from the connected devices.

The XML currently contains the following information:

- File information
- Location information
- Device information
- Visitor counting values (Hourly)
- Visitor counter obstructions
- Alarms
- Deactivations (Hourly)
- Detachments (Hourly)
- Point-of-Sale data (Hourly)

3.5.1 Future additions

It is possible that modifications will be made to the Cross Point Data Format. For this reason the **FileInfo** element contains a version number that indicates which data format (and schema) is used. At this moment the version of the Cross Point Data Format is **4.0**.

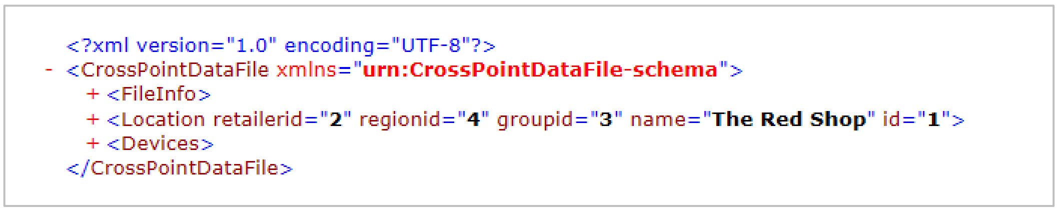
Note that when new elements are added to the XML the version number will not be increased, but when existing elements are changed the version number will be increased.

Cross Point recommends integrators to keep this in mind while integrating the XML data format.

3.5.2 Cross Point Data File (Root Element)

Each XML document contains one root element where all information is related to. For the Cross Point XML format this root element is called **CrossPointDataFile**.

The root element contains all the information about the controller and what is connected to it. In the following paragraphs each item will be handled.

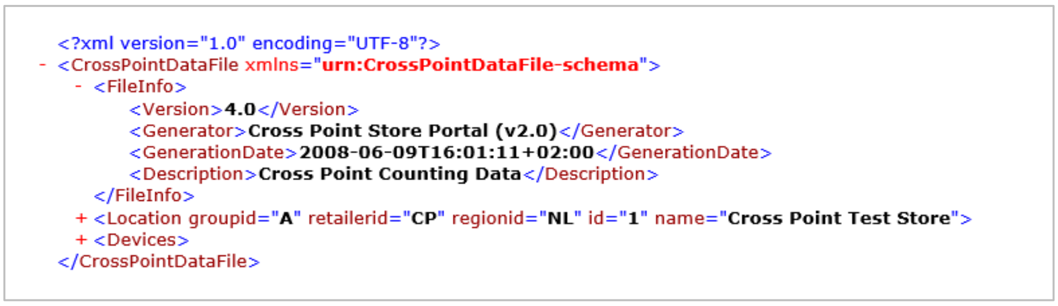


```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  + <FileInfo>
  + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
  + <Devices>
  </CrossPointDataFile>
```

Figure 2: CrossPointDataFile element example

3.5.3 The FileInfo element

The **FileInfo** element contains information about the Cross Point XML file such as generator, date and time of generation etc.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  - <FileInfo>
      <Version>4.0</Version>
      <Generator>Cross Point Store Portal (v2.0)</Generator>
      <GenerationDate>2008-06-09T16:01:11+02:00</GenerationDate>
      <Description>Cross Point Counting Data</Description>
    </FileInfo>
  + <Location groupid="A" retailerid="CP" regionid="NL" id="1" name="Cross Point Test Store">
  + <Devices>
  </CrossPointDataFile>
```

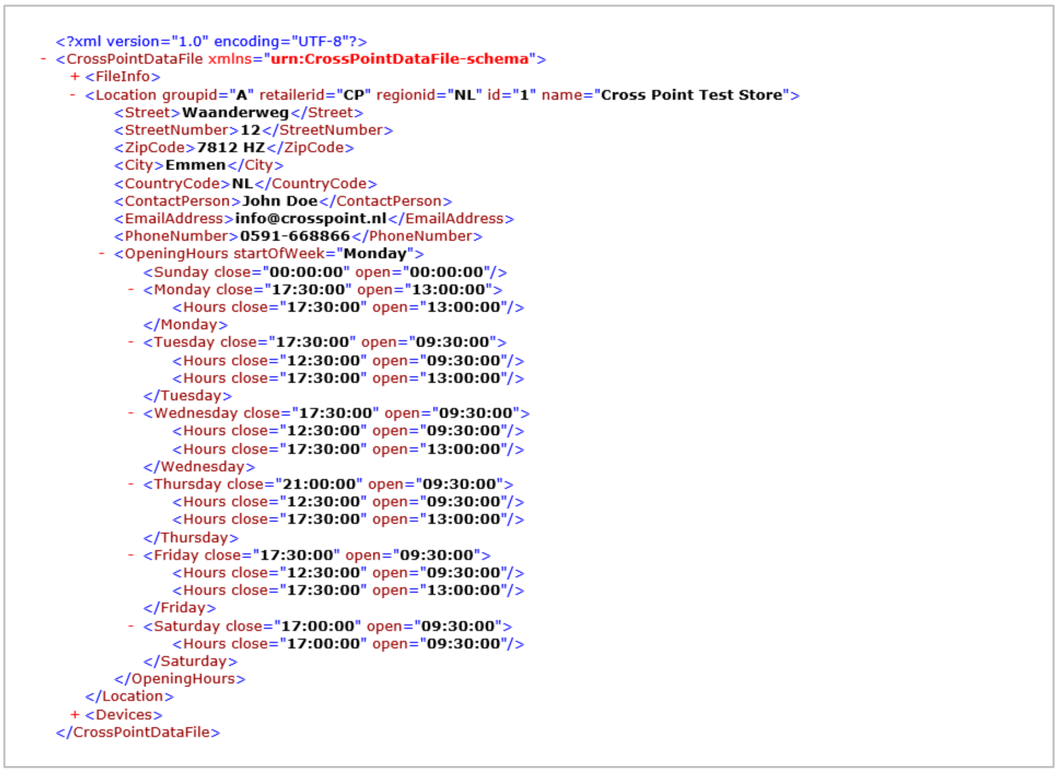
Figure 3: FileInfo element example

3.5.4 The Location element

The **Location** element contains information about the Cross Point Access Point and about the retail store where the Access Point is located.

The **OpeningHours** element contains the opening hours as configured in the Store Portal application. For each day it specifies the periods that the store is opened.

- When a store has multiple opening periods, each period will be listed separately. (In the example below the store is mostly opened from 9:30 to 17:30 but on Saturday from 9.30 to 12:00 and from 13:00 to 17:00 on Saturday)
- When a store is opened 24 hours a day, the open-time will be 0:00 and the close-time will be 23:59.
- When a store is closed the whole day there is no Hours element present and the open and close times are both 0:00



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  + <FileInfo>
  - <Location groupid="A" retailerid="CP" regionid="NL" id="1" name="Cross Point Test Store">
      <Street>Waanderweg</Street>
      <StreetNumber>12</StreetNumber>
      <ZipCode>7812 HZ</ZipCode>
      <City>Emmen</City>
      <CountryCode>NL</CountryCode>
      <ContactPerson>John Doe</ContactPerson>
      <EmailAddress>info@crosspoint.nl</EmailAddress>
      <PhoneNumber>0591-668866</PhoneNumber>
    - <OpeningHours startOfWeek="Monday">
        <Sunday close="00:00:00" open="00:00:00"/>
      - <Monday close="17:30:00" open="13:00:00">
          <Hours close="17:30:00" open="13:00:00"/>
        </Monday>
      - <Tuesday close="17:30:00" open="09:30:00">
          <Hours close="12:30:00" open="09:30:00"/>
          <Hours close="17:30:00" open="13:00:00"/>
        </Tuesday>
      - <Wednesday close="17:30:00" open="09:30:00">
          <Hours close="12:30:00" open="09:30:00"/>
          <Hours close="17:30:00" open="13:00:00"/>
        </Wednesday>
      - <Thursday close="21:00:00" open="09:30:00">
          <Hours close="12:30:00" open="09:30:00"/>
          <Hours close="17:30:00" open="13:00:00"/>
        </Thursday>
      - <Friday close="17:30:00" open="09:30:00">
          <Hours close="12:30:00" open="09:30:00"/>
          <Hours close="17:30:00" open="13:00:00"/>
        </Friday>
      - <Saturday close="17:00:00" open="09:30:00">
          <Hours close="17:00:00" open="09:30:00"/>
        </Saturday>
      </OpeningHours>
    </Location>
  + <Devices>
  </CrossPointDataFile>
```

Figure 4: OpeningHours element example

3.5.5 The Visitors element

Visitors are the number of persons that walked through the NEXUS antennas as counted by the visitor counter in the top of the NEXUS antennas.

The **Visitors** element contains the number of visitors for this location over a specific period. The visitors are divided into days, periods, entrances and aisles.

An entrance is a collection of aisles. An aisle is a combination of for example a Nexus Transceiver and Receiver:

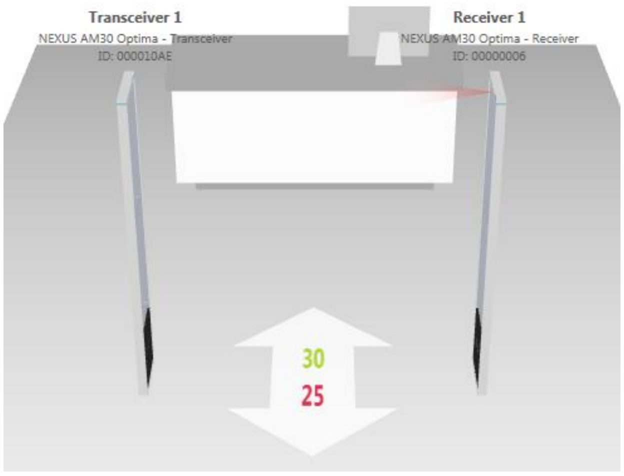


Figure 5: Aisle view with visitor counting in Device Explorer

Example The number of visitors can be used to calculate the conversion rate. The conversion rate is calculated using the following formula:

Conversion rate (%) = (Number of transactions / Number of visitors) x 100

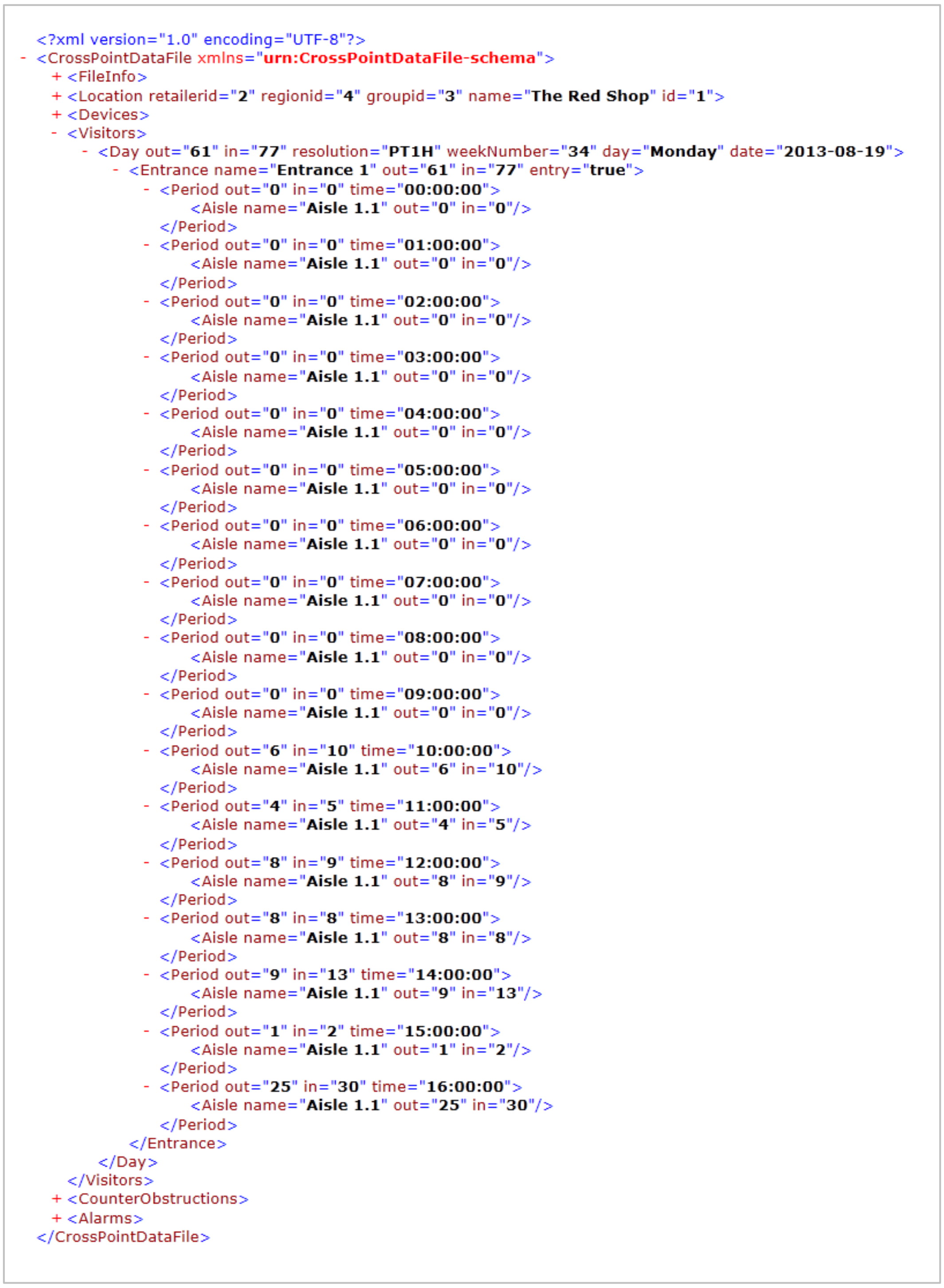
Within the **Visitors** element it is possible that there are multiple day elements.

Each element within the **Visitors** element contains the cumulative incoming (in) and outgoing (out) visitor counting's for that element. (The entrance contains 69 incoming and 63 outgoing visitors for that specific entrance) For the total number of visitors of each day it is possible to read out only the incoming and outgoing attribute for that day element.

Each day contains the time-resolution of that day. E.g. the current day has a resolution of 1 hour. In the example below this is defined as PT1H which is duration defined as **P<date>T<time>**

For more information about ISO time specifications check out Wikipedia at: http://en.wikipedia.org/wiki/ISO_8601

NOTE The **Visitors** element is only present in the XML file when there are systems connected to the CrossCONNECT Access Point with visitor counters enabled.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
   + <FileInfo>
   + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
   + <Devices>
   - <Visitors>
      - <Day out="61" in="77" resolution="PT1H" weekNumber="34" day="Monday" date="2013-08-19">
         - <Entrance name="Entrance 1" out="61" in="77" entry="true">
            - <Period out="0" in="0" time="00:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="01:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="02:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="03:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="04:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="05:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="06:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="07:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="08:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="0" in="0" time="09:00:00">
                 <Aisle name="Aisle 1.1" out="0" in="0"/>
              </Period>
            - <Period out="6" in="10" time="10:00:00">
                 <Aisle name="Aisle 1.1" out="6" in="10"/>
              </Period>
            - <Period out="4" in="5" time="11:00:00">
                 <Aisle name="Aisle 1.1" out="4" in="5"/>
              </Period>
            - <Period out="8" in="9" time="12:00:00">
                 <Aisle name="Aisle 1.1" out="8" in="9"/>
              </Period>
            - <Period out="8" in="8" time="13:00:00">
                 <Aisle name="Aisle 1.1" out="8" in="8"/>
              </Period>
            - <Period out="9" in="13" time="14:00:00">
                 <Aisle name="Aisle 1.1" out="9" in="13"/>
              </Period>
            - <Period out="1" in="2" time="15:00:00">
                 <Aisle name="Aisle 1.1" out="1" in="2"/>
              </Period>
            - <Period out="25" in="30" time="16:00:00">
                 <Aisle name="Aisle 1.1" out="25" in="30"/>
              </Period>
           </Entrance>
        </Day>
     </Visitors>
   + <CounterObstructions>
   + <Alarms>
  </CrossPointDataFile>
```

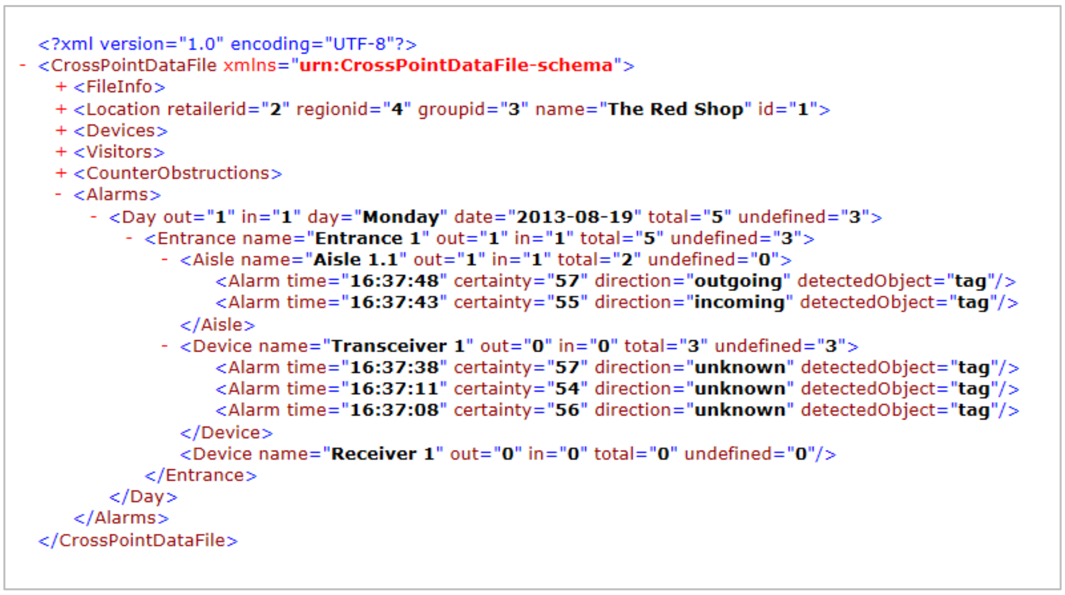
Figure 6: Visitors element example

3.5.6 The Alarms element

The **Alarms** element contains information about the number of alarms that occurred per aisle. Each alarm is listed separately with the corresponding information such as frequency, signal strength and number of detections for that alarm.

Note that it is possible that there are multiple days, entrances, aisles or alarms listed within the **Alarms** element.

Each element under the **Alarms** element contains the cumulative number of alarms.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  + <FileInfo>
  + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
  + <Devices>
  + <Visitors>
  + <CounterObstructions>
  - <Alarms>
    - <Day out="1" in="1" day="Monday" date="2013-08-19" total="5" undefined="3">
      - <Entrance name="Entrance 1" out="1" in="1" total="5" undefined="3">
        - <Aisle name="Aisle 1.1" out="1" in="1" total="2" undefined="0">
            <Alarm time="16:37:48" certainty="57" direction="outgoing" detectedObject="tag"/>
            <Alarm time="16:37:43" certainty="55" direction="incoming" detectedObject="tag"/>
          </Aisle>
        - <Device name="Transceiver 1" out="0" in="0" total="3" undefined="3">
            <Alarm time="16:37:38" certainty="57" direction="unknown" detectedObject="tag"/>
            <Alarm time="16:37:11" certainty="54" direction="unknown" detectedObject="tag"/>
            <Alarm time="16:37:08" certainty="56" direction="unknown" detectedObject="tag"/>
          </Device>
          <Device name="Receiver 1" out="0" in="0" total="0" undefined="0"/>
        </Entrance>
      </Day>
    </Alarms>
  </CrossPointDataFile>
```

Figure 7: Alarms element example

The **Alarms** element contains the following information about the alarm that occurred:

- The **time** attribute indicates the time at which the alarm occurred.
- The **direction** attribute indicates the type of alarm. This can be one of the following types:
  - Incoming: The system detected a tag while a visitor walks into the store.
  - Outgoing: The system detected a tag while a visitor walks out of the store.
  - Unknown: The system detected a tag but no movement between the antennas was detected.
- The **detectedObject** indicates what type of object has been detected. Possible values are tag and metal.
- The **certainty** attribute can be a value between 1 and 100 and indicates with how much certainty the generated alarm was a valid alarm. (AM systems only)
- The **frequency** attribute indicates the frequency on which the alarm occurred. The frequency is indicated in kHz. (RF systems only)
- The **signalstrength** attribute indicates the difference between the threshold point and the tag signal. The signal strength is indicated in mV. (RF systems only)
- The **thresholdpoint** attribute indicates the threshold level where the alarm occurred. The threshold point is indicated in mV. (RF systems only)

- The **detections** attribute indicates the number of detections within a specified amount of time. This can be configured in the Store Portal application. If multiple alarms occur within a small amount of time, they are merged into one alarm with multiple detections. (RF systems only)

3.5.7 The Deactivations element

A deactivation is a successful or failed attempt to blow-up a paper label (also called a tag) when a product is purchased at a checkout.

The **Deactivations** element contains information about the deactivations that took place at the several points of sale.

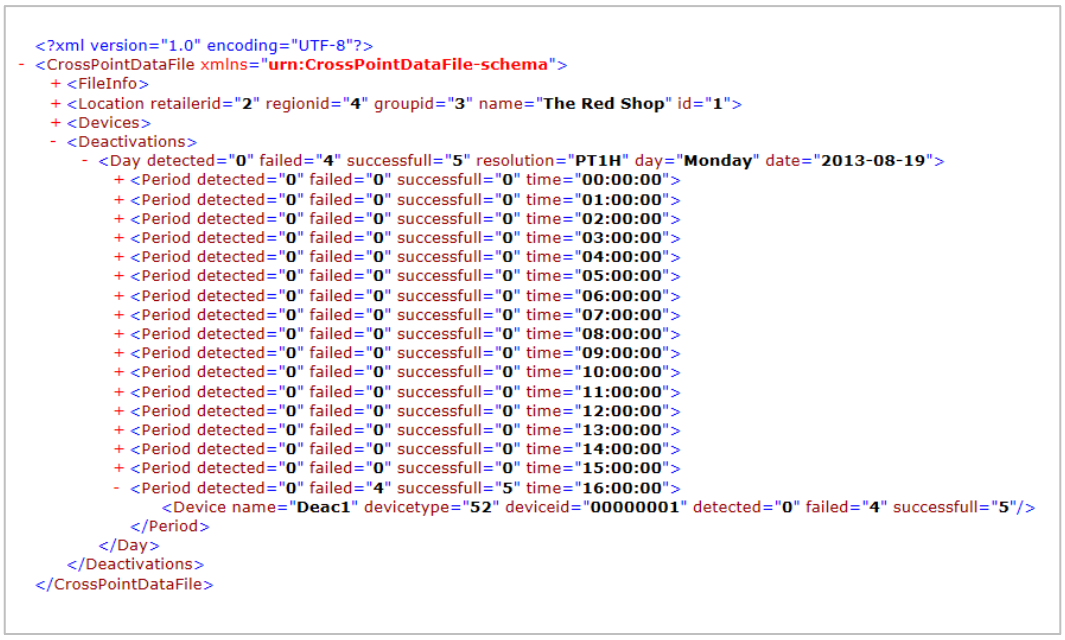
In contrast to the visitor counting and alarms, the deactivations are not divided into entrances and aisles because they are not part of an aisle. For this reason each deactivator and its successful and failed deactivations are listed per day and period.

Each item under the deactivations contains the cumulative deactivations. So the Day element contains the successful and failed deactivations of that complete day.

The resolution attribute of the day element contains the resolution that is used for the periods for that day. The resolution is a duration that is defined in the ISO time format.

For more information about the ISO format, check out Wikipedia: http://en.wikipedia.org/wiki/ISO_8601.

NOTE Deactivations are only present when one or more NEXUS Deactivators are connected to the CrossCONNECT Access Point.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
    + <FileInfo>
    + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
    + <Devices>
    - <Deactivations>
        - <Day detected="0" failed="4" successfull="5" resolution="PT1H" day="Monday" date="2013-08-19">
            + <Period detected="0" failed="0" successfull="0" time="00:00:00">
            + <Period detected="0" failed="0" successfull="0" time="01:00:00">
            + <Period detected="0" failed="0" successfull="0" time="02:00:00">
            + <Period detected="0" failed="0" successfull="0" time="03:00:00">
            + <Period detected="0" failed="0" successfull="0" time="04:00:00">
            + <Period detected="0" failed="0" successfull="0" time="05:00:00">
            + <Period detected="0" failed="0" successfull="0" time="06:00:00">
            + <Period detected="0" failed="0" successfull="0" time="07:00:00">
            + <Period detected="0" failed="0" successfull="0" time="08:00:00">
            + <Period detected="0" failed="0" successfull="0" time="09:00:00">
            + <Period detected="0" failed="0" successfull="0" time="10:00:00">
            + <Period detected="0" failed="0" successfull="0" time="11:00:00">
            + <Period detected="0" failed="0" successfull="0" time="12:00:00">
            + <Period detected="0" failed="0" successfull="0" time="13:00:00">
            + <Period detected="0" failed="0" successfull="0" time="14:00:00">
            + <Period detected="0" failed="0" successfull="0" time="15:00:00">
            - <Period detected="0" failed="4" successfull="5" time="16:00:00">
                  <Device name="Deac1" devicetype="52" deviceid="00000001" detected="0" failed="4" successfull="5"/>
              </Period>
          </Day>
      </Deactivations>
  </CrossPointDataFile>
```

Figure 8: Deactivations element example

3.5.8 The Detachements element

A detachment is the release of a security tag from a product at a checkout after it has been purchased.

The Detachements element contains information about the detachments that took place at the several points of sale. Please note that the element name is written with an extra letter “e”; Detach<u>e</u>ments

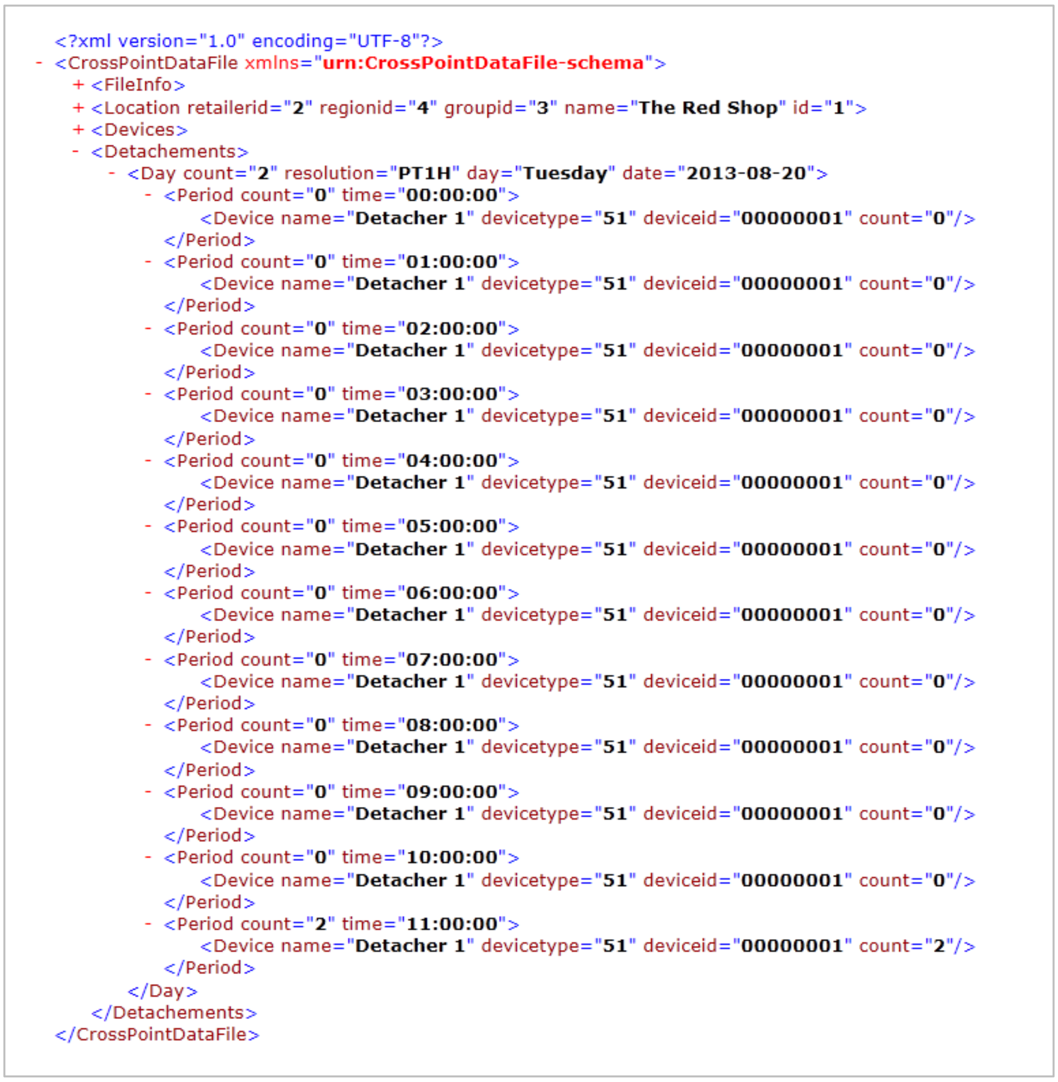
In contrast to the visitor counting and alarms, the detachments are not divided into entrances and aisles because they are not part of an aisle. For this reason each detacher and its detachments are listed per day and period.

Each item under the **Detachements** element contains the cumulative detachments. So the Day element contains the detachments of that complete day.

The resolution attribute of the day element contains the resolution that is used for the periods for that day. The resolution is a duration that is defined in the ISO time format.

For more information about the ISO format, check out Wikipedia: http://en.wikipedia.org/wiki/ISO_8601.

NOTE Detachments are only present when one or more NEXUS Detachers are connected to the CrossCONNECT Access Point.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
   + <FileInfo>
   + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
   + <Devices>
   - <Detachements>
      - <Day count="2" resolution="PT1H" day="Tuesday" date="2013-08-20">
         - <Period count="0" time="00:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="01:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="02:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="03:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="04:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="05:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="06:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="07:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="08:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="09:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="0" time="10:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="0"/>
           </Period>
         - <Period count="2" time="11:00:00">
              <Device name="Detacher 1" devicetype="51" deviceid="00000001" count="2"/>
           </Period>
        </Day>
     </Detachements>
  </CrossPointDataFile>
```

Figure 9: Detachements element example

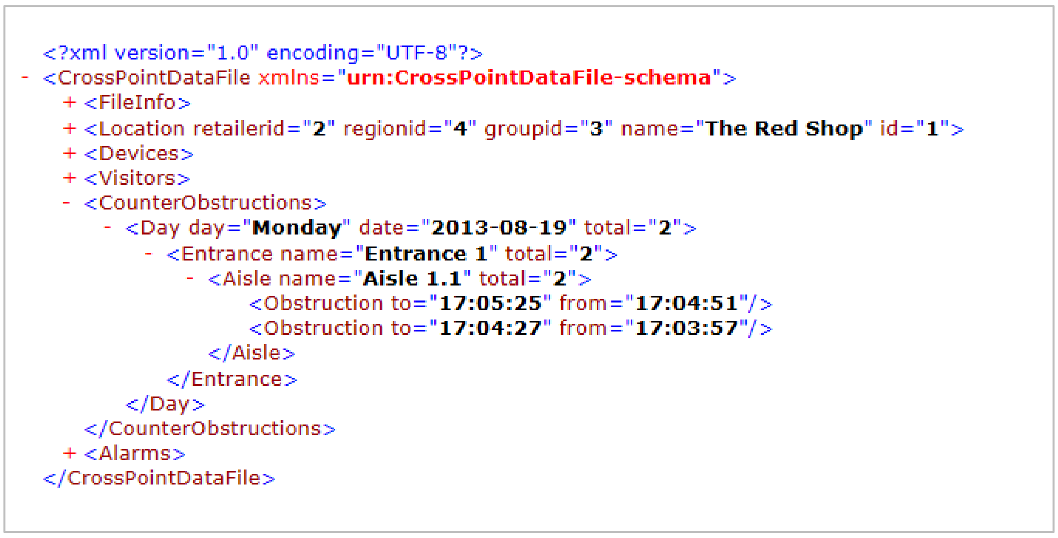
3.5.9 The CounterObstructions element

Counter obstructions are situations where an aisle is blocked for a while. After being blocked for more than 15 seconds (default value), the NEXUS device notifies the CrossCONNECT Access Point that it is being blocked. When an aisle is blocked for a larger amount of time, the counting values within that time block aren't accurate anymore.

The **CounterObstructions** element contains a listing of all obstructions that took place at the entrances and aisles at the current location.

Each obstruction is divided into the aisle and entrance where it took place. For each element under the **CounterObstructions** element, the number of obstructions are listed. In the example below there were 2 obstructions on Monday.

The **total** attribute is the cumulative number of obstructions for that specific element. To use the number of obstructions for the day it is possible to read the total attribute on the day element instead of reading the complete tree.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  + <FileInfo>
  + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
  + <Devices>
  + <Visitors>
  - <CounterObstructions>
    - <Day day="Monday" date="2013-08-19" total="2">
      - <Entrance name="Entrance 1" total="2">
        - <Aisle name="Aisle 1.1" total="2">
            <Obstruction to="17:05:25" from="17:04:51"/>
            <Obstruction to="17:04:27" from="17:03:57"/>
          </Aisle>
        </Entrance>
      </Day>
    </CounterObstructions>
  + <Alarms>
  </CrossPointDataFile>
```

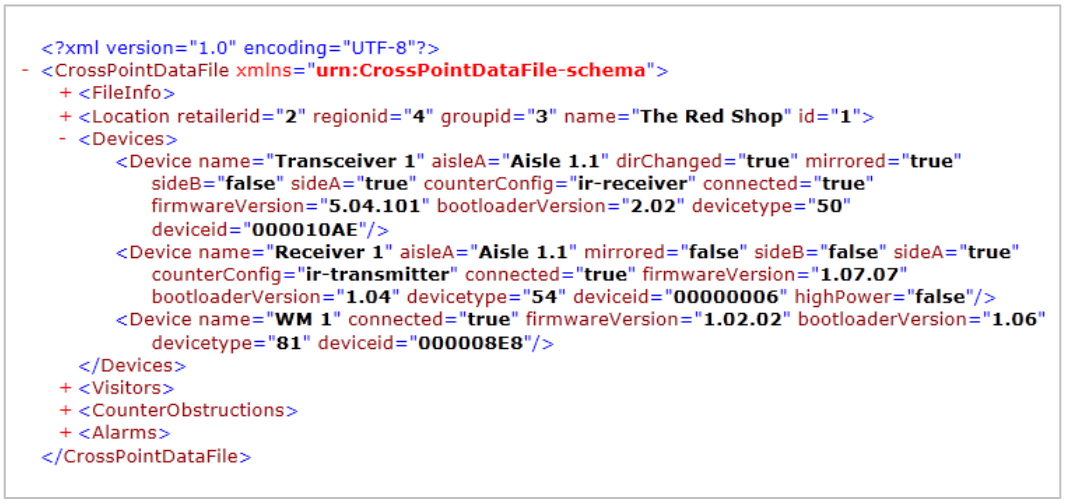
Figure 10: CounterObstructions element example

The **CounterObstructions** element contains the **from** and **to** attributes, which indicate the start and end of the obstruction.

3.5.10 The Devices element

The **Devices** element gives an overview of the devices that are known by the CrossCONNECT Access Point. Each device, whether or not connected to the Access Point is listed in this overview.

When an arrangement is accomplished, the aisle where a device is part of is also listed.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  + <FileInfo>
  + <Location retailerid="2" regionid="4" groupid="3" name="The Red Shop" id="1">
  - <Devices>
      <Device name="Transceiver 1" aisleA="Aisle 1.1" dirChanged="true" mirrored="true"
        sideB="false" sideA="true" counterConfig="ir-receiver" connected="true"
        firmwareVersion="5.04.101" bootloaderVersion="2.02" devicetype="50"
        deviceid="000010AE"/>
      <Device name="Receiver 1" aisleA="Aisle 1.1" mirrored="false" sideB="false" sideA="true"
        counterConfig="ir-transmitter" connected="true" firmwareVersion="1.07.07"
        bootloaderVersion="1.04" devicetype="54" deviceid="00000006" highPower="false"/>
      <Device name="WM 1" connected="true" firmwareVersion="1.02.02" bootloaderVersion="1.06"
        devicetype="81" deviceid="000008E8"/>
    </Devices>
  + <Visitors>
  + <CounterObstructions>
  + <Alarms>
  </CrossPointDataFile>
```

Figure 11: Devices element example

The Device element contains the following attributes:

- name: The name of the device.
- deviceid: The unique ID of the device.
- devicetype: The type of the device. Can be one of the following:
  - 48: NEXUS RF 24Gi Transmitter
  - 49: NEXUS RF 24Gi Receiver
  - 50: NEXUS AM Transceiver
  - 51: NEXUS RF Detacher
  - 52: NEXUS RF Deactivator
  - 54: NEXUS AM Receiver
  - 55: NEXUS AM Deactivator
  - 57: NEXUS RF 25Gi Transmitter
  - 58: NEXUS RF 25Gi Receiver
  - 80: Wireless Receiver/Repeater
  - 81: Wireless Receiver/Repeater Extension Board
  - 82: Standalone Visitor Counter
  - 96: MAXUS AM Transceiver
  - 97: MAXUS AM Receiver
- bootloaderVersion: The version of the bootloader inside the device.
- firmwareVersion: The version of the firmware inside the device.
- connected: Whether or not the device is connected to the Access Point.

The following elements are optional and device specific.

- sideA: Whether or not the visitor counter on the A side of the device is enabled.
- sideB: Whether or not the visitor counter on the B side of the device is enabled.
- dirChanged: Whether or not the directions are toggled (in/out).
- mirrored: Whether or not the device is swapped (the device is on the left or right side).
- highpower: Whether or not the high power setting is used.
- counterConfig: Whether the counter is a transmitter or receiver.

3.5.11 The Sales element

To integrate Point-of-Sale data in Cross Point applications like Analytics, this data can be added to the XML file which can be sent to the Cross Point Cloud.

The Sales element contains information about the transactions that took place at the several checkouts.

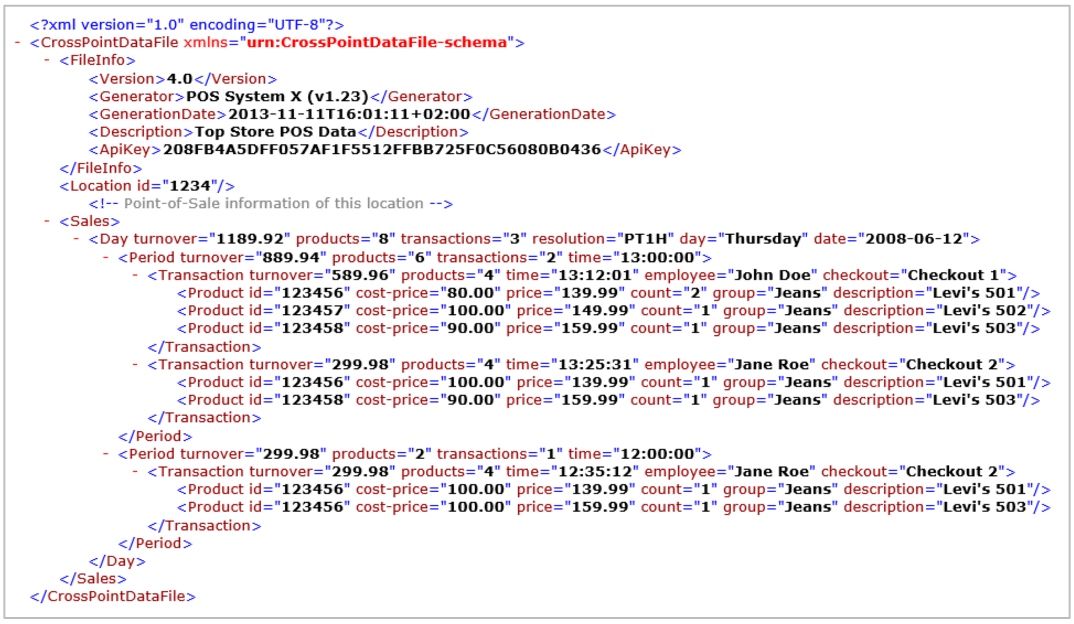
In contrast to the visitor counting and alarms, the transactions are not divided into entrances and aisles because they are not part of an aisle. For this reason each transaction of a check-out is listed per day and period.

Each item under the **Sales** element contains the cumulative products, transactions and turnover. So the Day element contains the number of transactions and products of that complete day.

The product price and cost-price attributes contain the value for a single product item. The count indicates the number of items purchased of this product within this transaction.

The resolution attribute of the day element contains the resolution that is used for the periods for that day. The resolution is a duration that is defined in the ISO time format.

For more information about the ISO format, check out Wikipedia: http://en.wikipedia.org/wiki/ISO_8601.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  - <FileInfo>
      <Version>4.0</Version>
      <Generator>POS System X (v1.23)</Generator>
      <GenerationDate>2013-11-11T16:01:11+02:00</GenerationDate>
      <Description>Top Store POS Data</Description>
      <ApiKey>208FB4A5DFF057AF1F5512FFBB725F0C56080B0436</ApiKey>
    </FileInfo>
    <Location id="1234"/>
      <!-- Point-of-Sale information of this location -->
  - <Sales>
    - <Day turnover="1189.92" products="8" transactions="3" resolution="PT1H" day="Thursday" date="2008-06-12">
      - <Period turnover="889.94" products="6" transactions="2" time="13:00:00">
        - <Transaction turnover="589.96" products="4" time="13:12:01" employee="John Doe" checkout="Checkout 1">
            <Product id="123456" cost-price="80.00" price="139.99" count="2" group="Jeans" description="Levi's 501"/>
            <Product id="123457" cost-price="100.00" price="149.99" count="1" group="Jeans" description="Levi's 502"/>
            <Product id="123458" cost-price="90.00" price="159.99" count="1" group="Jeans" description="Levi's 503"/>
          </Transaction>
        - <Transaction turnover="299.98" products="4" time="13:25:31" employee="Jane Roe" checkout="Checkout 2">
            <Product id="123456" cost-price="100.00" price="139.99" count="1" group="Jeans" description="Levi's 501"/>
            <Product id="123458" cost-price="90.00" price="159.99" count="1" group="Jeans" description="Levi's 503"/>
          </Transaction>
        </Period>
      - <Period turnover="299.98" products="2" transactions="1" time="12:00:00">
        - <Transaction turnover="299.98" products="4" time="12:35:12" employee="Jane Roe" checkout="Checkout 2">
            <Product id="123456" cost-price="100.00" price="139.99" count="1" group="Jeans" description="Levi's 501"/>
            <Product id="123456" cost-price="100.00" price="159.99" count="1" group="Jeans" description="Levi's 503"/>
          </Transaction>
        </Period>
      </Day>
    </Sales>
  </CrossPointDataFile>
```

Figure 12: Sales element example

3.6 Cross Point XML Schema

As already discussed, XML documents must be written well formed. To verify this, an XML scheme can be used.

An XML document that complies with a particular schema/DTD, in addition to being well-formed, is said to be valid.

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic constraints imposed by XML itself. A number of standard and proprietary XML schema languages have emerged for the purpose of formally expressing such schemas, and some of these languages are XML-based, themselves.

Cross Point delivers a schema to be able to validate a generated XML. This schema can also be useful for other users of the Cross Point XML file format.

3.7 Retrieving the XML from the Access Point

3.7.1 Introduction

The CrossCONNECT Access Point is able to generate XML output to enable other systems to process this information.

In this chapter a short explanation is given about the several ways to receive the XML output from the CrossCONNECT Access Point.

Mainly there are two ways to get the data from the Access Point:

1. The Access Point pushes data to another system (Email, (S)FTP, File sharing)
2. Another system pulls data from the Access Point (HTTP request, (S)FTP)

3.7.2 Push data to another system

The first and easiest way to receive data from the Access Point is to send the data periodically to another system. This can be done by creating a report in the Reports page and send it automatically using the following methods:

- By email
- By sending over Windows File Sharing, FTP (File Transfer Protocol) or SFTP

Both ways will be described in the following paragraphs.

When pushing data to other systems, the time that the generation takes place must be configured. For each report the time and recursion can be configured. The recursion is one of the following:

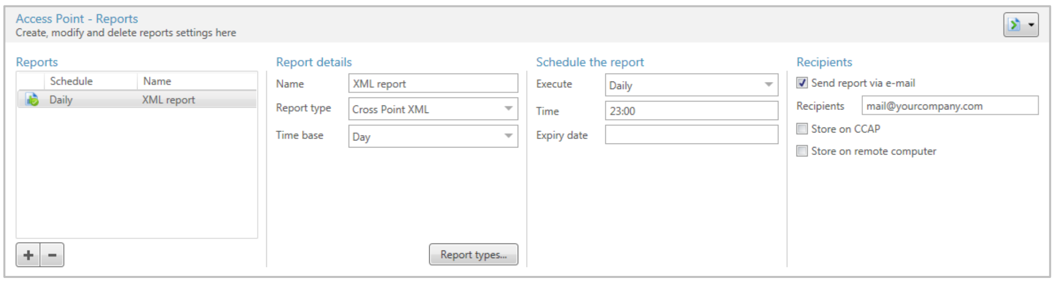
- Each 15 minutes
- Each 30 minutes
- Hourly
- Daily
- Weekly
- Monthly

When the recursion is longer than one day, the report that is generated automatically contains information from the previous days that weren't scheduled. (Example: When a report is sent on Wednesday and Saturday, the report of Saturday contains information from Thursday till Saturday.)

3.7.2.1 By email

A report can be sent by email when email addresses are entered in the recipient's field of the report. The XML data is then attached to this email.

In the following picture an example of a daily report is given where the XML file is sent daily at 23:00 to mail@yourcompany.com



Access Point - Reports
Create, modify and delete reports settings here

Reports

Schedule	Name
Daily	XML report

Report details

Name	XML report
Report type	Cross Point XML
Time base	Day

Schedule the report

Execute	Daily
Time	23:00
Expiry date	

Recipients

☑ Send report via e-mail
Recipients: mail@yourcompany.com
☐ Store on CCAP
☐ Store on remote computer

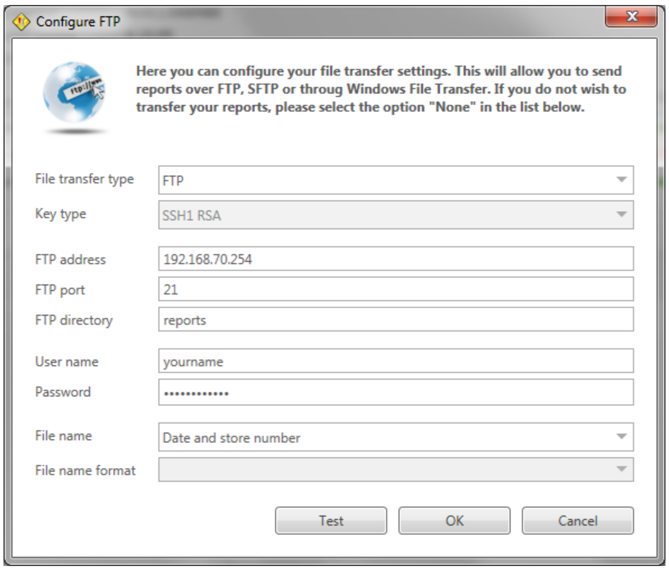
Figure 13: Email report configuration example in Device Explorer

3.7.2.2 By FTP

A report can be sent to another FTP server. To do so, check the "Store on remote computer" option in the report and fill in the FTP address, username and password and optionally a remote directory where the files are placed on the server.

NOTE A configured FTP server is needed on the receiving computer.

In the following example the XML file will be transferred via FTP to the system with an IP address. The file will be stored in the Reports folder and a username and password must be supplied for authentication on the FTP server.



Configure FTP

Here you can configure your file transfer settings. This will allow you to send reports over FTP, SFTP or throug Windows File Transfer. If you do not wish to transfer your reports, please select the option "None" in the list below.

File transfer type	FTP
Key type	SSH1 RSA
FTP address	192.168.70.254
FTP port	21
FTP directory	reports
User name	yourname
Password	••••••••••••
File name	Date and store number
File name format	

Figure 14: FTP configuration window in Device Explorer

3.7.3 Pull data from the Access Point

The second way to receive data is to pull the data from the Access Point. This can be done using the following methods:

- Generating the data periodically and download it over FTP.
- Performing an HTTP request and generate the data on the fly.

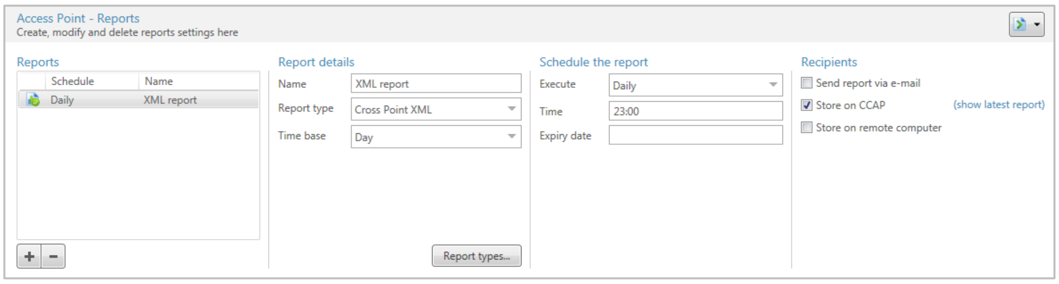
Both ways will be described in the following paragraphs.

3.7.4 Generating data and downloading over FTP

To pull data from the Access Point over FTP, the file that will be downloaded must be generated at a certain interval. For more information about scheduling of reports, please look at the previous paragraph where this is explained.

When a file must be generated and be available for downloading over FTP, the following settings have to be activated:

- Activate the “Store on CCAP” option in the Recipients section of the Reports screen.
- Activate the “Allow FTP access” option in the Network settings screen.



Access Point - Reports
Create, modify and delete reports settings here

Reports

Schedule	Name
Daily	XML report

Report details

Name	XML report
Report type	Cross Point XML
Time base	Day

Schedule the report

Execute	Daily
Time	23:00
Expiry date	

Recipients

- ☐ Send report via e-mail
- ☑ Store on CCAP (show latest report)
- ☐ Store on remote computer

Figure 15: Store report on CCAP configuration example in Device Explorer

The actual report file is placed on the Access Point in the Reports folder and will be overwritten each time the report is generated on the same day.

The Access Point will be accessible through FTP with the following credentials:

- Username : anonymous
- Password : report

3.7.5 Using an HTTP request

The second way to pull data from the Access Point is with an HTTP request. This request is sent to the web server of the Access Point and will be processed on-the-fly.

3.7.5.1 Get statistics based on a report

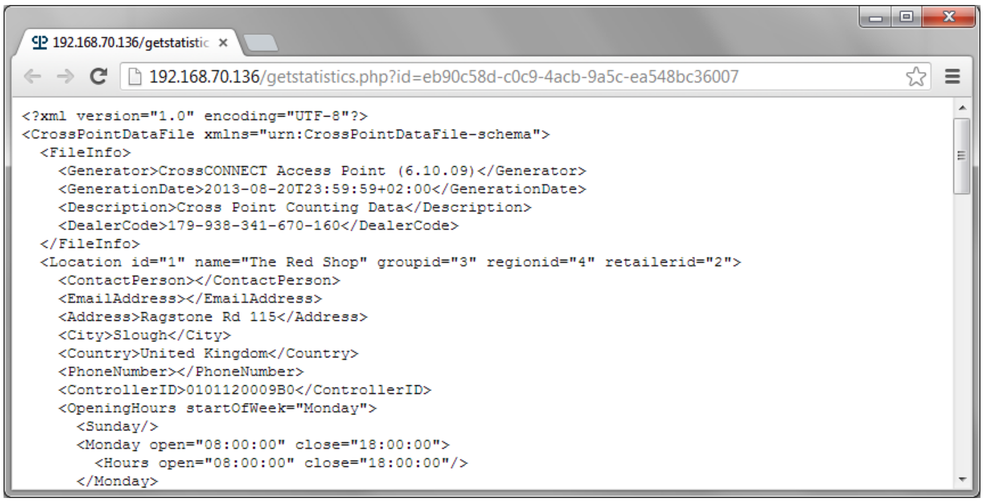
To retrieve the data directly from the Access Point, a PHP page on the Access Point is used. The URL of the request can be entered into the address bar of your browser window.

To obtain the URL, a report must be created first using Device Explorer. Follow the next steps to create and generate a report:

- Go to the Reports page and define the report to be stored on the CCAP,
- Create the report by clicking on the “Generate report for today” button
- Click on the “Show latest report” link to view the report. Your browser will open and the report content is displayed.

After completing the above steps, the URL for this HTTP request is now displayed in the address field of your browser and should look like the following example:

http://192.168.70.136/getstatistics.php?id=eb90c58d-c0c9-4acb-9a5c-ea548bc36007


192.168.70.136/getstatistic
192.168.70.136/getstatistics.php?id=eb90c58d-c0c9-4acb-9a5c-ea548bc36007

```
<?xml version="1.0" encoding="UTF-8"?>
<CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  <FileInfo>
    <Generator>CrossCONNECT Access Point (6.10.09)</Generator>
    <GenerationDate>2013-08-20T23:59:59+02:00</GenerationDate>
    <Description>Cross Point Counting Data</Description>
    <DealerCode>179-938-341-670-160</DealerCode>
  </FileInfo>
  <Location id="1" name="The Red Shop" groupid="3" regionid="4" retailerid="2">
    <ContactPerson></ContactPerson>
    <EmailAddress></EmailAddress>
    <Address>Ragstone Rd 115</Address>
    <City>Slough</City>
    <Country>United Kingdom</Country>
    <PhoneNumber></PhoneNumber>
    <ControllerID>0101120009B0</ControllerID>
    <OpeningHours startOfWeek="Monday">
      <Sunday/>
      <Monday open="08:00:00" close="18:00:00">
        <Hours open="08:00:00" close="18:00:00"/>
      </Monday>
```

Figure 16: XML report created with the HTTP request

The IP address in the above example is the actual IP address of the Access Point, which has either been set manually (by you) or by the DHCP server in your local network. Make sure to replace this IP address with the actual value of your Access Point.

Also the report ID in this example is a unique ID. Once you have generated your report once (following the above mentioned steps), you will see the unique ID of your report in the URL.

This URL can then be used in your application to pull the report from the Access Point.

NOTE Try to limit the number of HTTP requests to the Access Point as much as possible. Generating files (and converting using XSLT) is an intensive task.

3.8 Sending the XML to the Cross Point Cloud

3.8.1 Introduction

To integrate POS data in Cross Point Analytics an XML file containing the data format described in this document can be send to the Cross Point Cloud. This data will be imported into the Cross Point cloud databases so it will be part of the available data for this retailer.

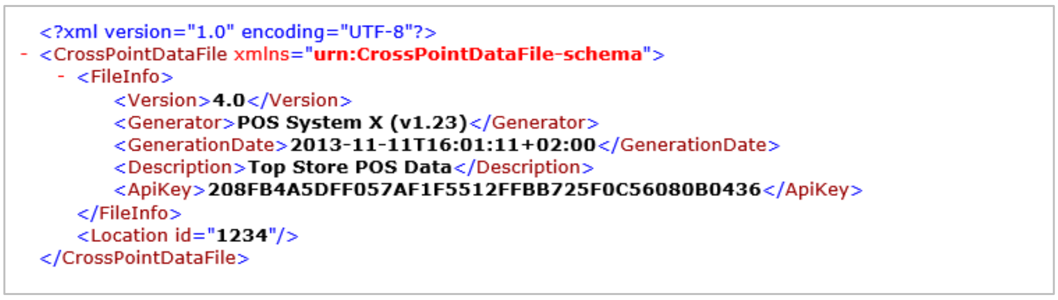
In this chapter a short explanation is given about how this data should be send to Cross Point.

3.8.2 API key

To be able to send data to the Cross Point Cloud, an API key is needed. This key will be used to validate the integrity of the sender of the data. The key must be added to the **FileInfo** element. Also the ID of the store is required. The store ID will be used to link the data in the file to the correct store entity in Cross Point Analytics.

NOTE The API key will be supplied by Cross Point or a Cross Point Partner.

NOTE The store ID which is used to link the data to a store can be configured in Analytics or in Device Explorer.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  - <FileInfo>
      <Version>4.0</Version>
      <Generator>POS System X (v1.23)</Generator>
      <GenerationDate>2013-11-11T16:01:11+02:00</GenerationDate>
      <Description>Top Store POS Data</Description>
      <ApiKey>208FB4A5DFF057AF1F5512FFBB725F0C56080B0436</ApiKey>
    </FileInfo>
    <Location id="1234"/>
  </CrossPointDataFile>
```

Figure 17: API key and store ID example

3.8.3 Sending data through FTP

Data can be sent to the Cross Point Cloud by using FTP or FTPS. FTPS is the secure variant of FTP. All data transfers are encrypted. The user name and password which need to be used to send the data will be supplied by Cross Point or a Cross Point Partner.

• FTP url:	**analytics.crosspoint.nl**
• FTP port:	**21** or **990**
• FTP encryption:	**None, explicit or implicit**

The format of the file name which must be used to store the file needs to be in a fixed format. This prevents that files will be overwritten when the same file name is used. The file name contains the date on which the file is generated and the store ID.

- File name format: **{YYYY}-{MM}-{DD} {hh}:{mm}:{ss} {ID}.xml**
    - {YYYY} The year as a four-digit number.
    - {MM} The month as a two-digit number.
    - {DD} The day as a two-digit number.
    - {hh} The hour as a two-digit number.
    - {mm} The minute as a two-digit number.
    - {ss} The seconds as a two-digit number.
    - {ID} The ID of the store
- Example: 2012-10-08 12:00:00 1234.xml

3.8.4 POS data file example

Below an example is given of a Cross Point Data File which contains information about the Point-of-Sale transactions of a store with ID 1234.

```
  <?xml version="1.0" encoding="UTF-8"?>
- <CrossPointDataFile xmlns="urn:CrossPointDataFile-schema">
  - <FileInfo>
        <Version>4.0</Version>
        <Generator>POS System X (v1.23)</Generator>
        <GenerationDate>2013-11-11T16:01:11+02:00</GenerationDate>
        <Description>Top Store POS Data</Description>
        <ApiKey>208FB4A5DFF057AF1F5512FFBB725F0C56080B0436</ApiKey>
    </FileInfo>
    <Location id="1234"/>
  - <Sales>
    - <Day turnover="1189.92" products="8" transactions="3" resolution="PT1H" day="Thursday" date="2008-06-12">
      - <Period turnover="889.94" products="6" transactions="2" time="13:00:00">
        - <Transaction turnover="589.96" products="4" time="13:12:01" employee="John Doe" checkout="Checkout 1">
              <Product id="123456" cost-price="80.00" price="139.99" group="Jeans" description="Levi's 501"/>
              <Product id="123457" cost-price="100.00" price="149.99" group="Jeans" description="Levi's 502"/>
              <Product id="123458" cost-price="90.00" price="159.99" group="Jeans" description="Levi's 503"/>
              <Product id="123456" cost-price="100.00" price="139.99" group="Jeans" description="Levi's 501"/>
          </Transaction>
        - <Transaction turnover="299.98" products="4" time="13:25:31" employee="Jane Roe" checkout="Checkout 2">
              <Product id="123456" cost-price="100.00" price="139.99" group="Jeans" description="Levi's 501"/>
              <Product id="123458" cost-price="90.00" price="159.99" group="Jeans" description="Levi's 503"/>
          </Transaction>
        </Period>
      - <Period turnover="299.98" products="2" transactions="1" time="12:00:00">
        - <Transaction turnover="299.98" products="4" time="12:35:12" employee="Jane Roe" checkout="Checkout 2">
              <Product id="123456" cost-price="100.00" price="139.99" group="Jeans" description="Levi's 501"/>
              <Product id="123456" cost-price="100.00" price="159.99" group="Jeans" description="Levi's 503"/>
          </Transaction>
        </Period>
      </Day>
    </Sales>
  </CrossPointDataFile>
```

Figure 18: Example of Point-of-Sale XML file

3.9 Converting Cross Point XML to other formats

3.9.1 Introduction

This chapter gives an explanation about the conversion possibilities of the Cross Point XML format.

The preferred way to receive the information from the CrossCONNECT Access Point is via the Cross Point XML format. Sometimes it is not possible to import this data format in to another system. In many of these situations the old format CSV is used. It is possible to convert the Cross Point XML format to this old CSV format, and many more text based formats.

WARNING When the data is converted from Cross Point XML to another format, it is possible that information such as alarms, deactivations and counter obstructions are lost! Also future information will be gone.

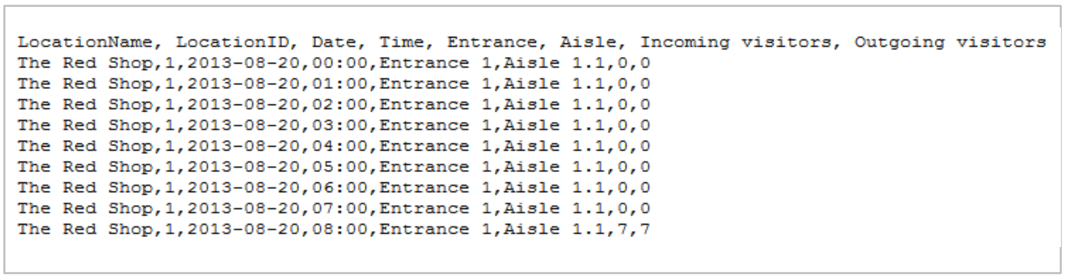
3.9.2 Programmer skills

A programmer who is going to develop a conversion script from the XML format to another format should meet the following skills:

- Have knowledge of the XML language.
- Have knowledge of the XSLT transformation language.
- Have knowledge of the Cross Point XML format.

3.9.3 Information about CSV

The comma separated values (CSV) is a data format that contains data that is delimited by a reserved character, mostly a comma but also semicolons or tab spaces.

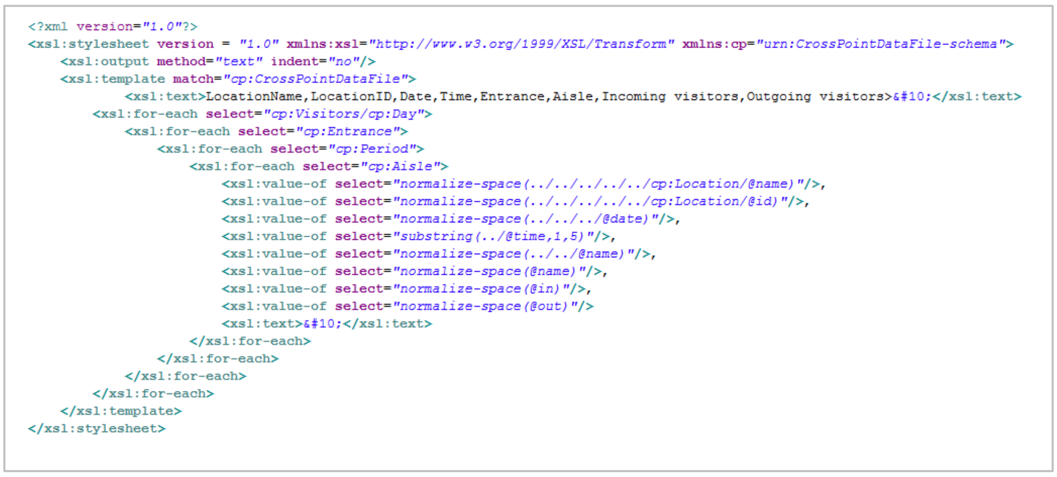


```
LocationName, LocationID, Date, Time, Entrance, Aisle, Incoming visitors, Outgoing visitors
The Red Shop,1,2013-08-20,00:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,01:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,02:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,03:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,04:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,05:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,06:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,07:00,Entrance 1,Aisle 1.1,0,0
The Red Shop,1,2013-08-20,08:00,Entrance 1,Aisle 1.1,7,7
```

Figure 19: CSV example

3.9.4 Example: Converting to CSV

In the following code example the XML format is converted to a custom CSV file using XSLT:



```
<?xml version="1.0"?>
<xsl:stylesheet version = "1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:cp="urn:CrossPointDataFile-schema">
    <xsl:output method="text" indent="no"/>
    <xsl:template match="cp:CrossPointDataFile">
            <xsl:text>LocationName,LocationID,Date,Time,Entrance,Aisle,Incoming visitors,Outgoing visitors>&#10;</xsl:text>
        <xsl:for-each select="cp:Visitors/cp:Day">
            <xsl:for-each select="cp:Entrance">
                <xsl:for-each select="cp:Period">
                    <xsl:for-each select="cp:Aisle">
                        <xsl:value-of select="normalize-space(../../../../../cp:Location/@name)"/>,
                        <xsl:value-of select="normalize-space(../../../../../cp:Location/@id)"/>,
                        <xsl:value-of select="normalize-space(../../../../@date)"/>,
                        <xsl:value-of select="substring(../@time,1,5)"/>,
                        <xsl:value-of select="normalize-space(../../@name)"/>,
                        <xsl:value-of select="normalize-space(@name)"/>,
                        <xsl:value-of select="normalize-space(@in)"/>,
                        <xsl:value-of select="normalize-space(@out)"/>
                        <xsl:text>&#10;</xsl:text>
                    </xsl:for-each>
                </xsl:for-each>
            </xsl:for-each>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

Figure 20: XSLT code example

The most important elements of the XSLT code example are:

- Apply the XSLT code to the XML root element (CrossPointDataFile)

  `<xsl:template match="cp:CrossPointDataFile">`

- Print the header of the CSV file once

  `<xsl:text>LocationName,LocationID,. . .&#10;</xsl:text>`

- Loop through each Day element, Entrance element, Period and Aisle element inside the Visitors element:

  `<xsl:for-each select="cp:Visitors/cp:Day">`

  `<xsl:for-each select="cp:Entrance">`

  `<xsl:for-each select="cp:Period">`

  `<xsl:for-each select="cp:Aisle">`

- Display the Store name and ID:

  `<xsl:value-of select="normalize-space(../../../../../cp:Location/@name)"/>`

  `<xsl:value-of select="normalize-space(../../../../../cp:Location/@id)"/>`

  Here the StoreName and StoreID are retrieved by walking back until the root element is reached, and then the location element is selected.

- Display the number of incoming visitors (from the current aisle element):

  `<xsl:value-of select="normalize-space(@in)"/>`

  Here the number of incoming visitors is displayed by retrieving the **in** attribute (using the @ sign)

- Create a new line after each data line (ASCII: 0x0A):

  `<xsl:text>&#10;</xsl:text>`

The above example XSLT code creates the output as displayed in Figure 19.

3.9.5 Converting the Cross Point XML

The Cross Point XML data can be formatted in two ways:

- **On a retailer IT system**

  This is done by using the Cross Point XML format and performing the transformation on the side of the retailer. This the preferred method.

- **Within the CrossCONNECT Access Point**

  This is done by uploading the transformation file on the Access Point and using the output in any other system.

3.9.5.1 Converting on a Retail IT system (Windows)

The msxsl.exe command line utility enables you to perform command line Extensible Stylesheet Language (XSL) transformations using the Microsoft XSL processor.

The XSLT utility (msxsl.exe) can be downloaded from the Microsoft website.

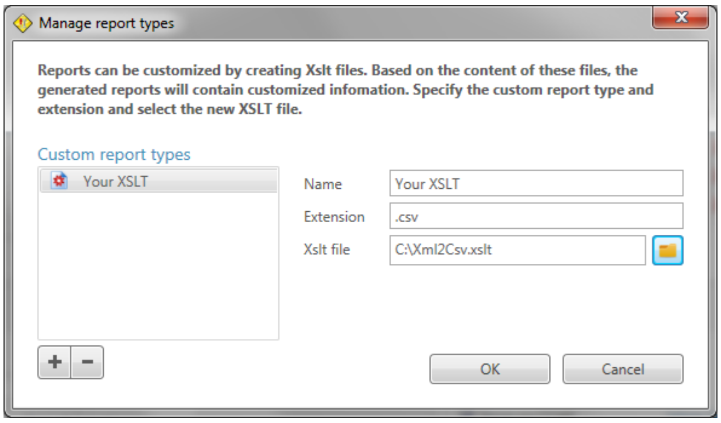
When the msxsl.exe utility is installed and also Microsoft XML 4.0 is installed on the system you can transform XML to any format using the following command line statement:

```
C:\msxsl.exe "CrossPoint.xml" "Your_XSLT.xslt" -o "Your_output_file.csv"
```

In this example the MSXSL utility processes the CrossPoint.xml input file with the provided XSLT file. This conversion results in the Your_output_file.csv output file.

3.9.5.2 Converting within the CrossCONNECT Access Point

In situations where conversion on the client side is not possible, it is possible to upload the self-made XSLT file to the Access Point using Device Explorer. The Access Point can convert the XML into the desired format and send it through the available options.



Reports can be customized by creating Xslt files. Based on the content of these files, the generated reports will contain customized infomation. Specify the custom report type and extension and select the new XSLT file.

Your XSLT

Name: Your XSLT
Extension: .csv
Xslt file: C:\Xml2Csv.xslt

Figure 21: Upload a custom report type

To upload a custom XSLT file to the Access Point using Device Explorer, open the Reports page and click on the **Report types…** button to open the Manage report types window. Enter the unique name for this report output type, the custom extension (e.g. CSV or TXT) and select the file name of the self-made XSLT file.

WARNING It is not recommended to test your XSLT files directly on the Access Point. We recommend testing it using a standalone XSLT utility such as Microsoft msxsl.exe.

4 Real-time data collection

Report based data collection, as described in the previous chapter, provides counting data with a shortest interval of 15 minutes. In case the data needs to be retrieved with a shorter interval or data needs to be collected immediately after the shop closes, real-time data collection must be used. An API has been created to enabled real-time data retrieval and the available commands are described in the next sections.

4.1 Get Counting Data

The command "getcountingdata", available on Access Point v7.03 and later, will enable the collection of the counting data of the current day. The request can be sent at any moment and will return the actual counting data up to that moment.

To use this feature, an HTTP request must be sent from a client computer, using the IP address or name of the Access Point followed by the api folder name and the "getcountingdata" command.

Example: http://<ipaddressccap>/api/getcountingdata

When the request is placed without attributes, the visitor counting data will be returned by default. In case other available counting data needs to be collected, the "type" attribute can be added to the request.

Example: http://<ipaddressccap>/api/getcountingdata?type=alarms

Available counting types are:

- visitors
- tag-alarms
- deactivations
- detachments
- metal-alarms
- detections[1]

1 - Only available in combination with the NEXUS AM Deactivator

4.1.1 Data output formats

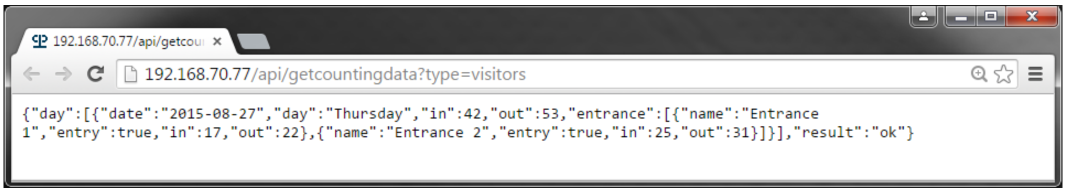
The getcountingdata command can result in the following data output formats:

- JSON
- XML

4.1.1.1 JSON

When the HTTP request is successful, the collected counting data is by default returned in JSON format.

Example:



```
{"day":[{"date":"2015-08-27","day":"Thursday","in":42,"out":53,"entrance":[{"name":"Entrance
1","entry":true,"in":17,"out":22},{"name":"Entrance 2","entry":true,"in":25,"out":31}]}],"result":"ok"}
```

Figure 22: JSON output created with the HTTP request

4.1.1.2 XML

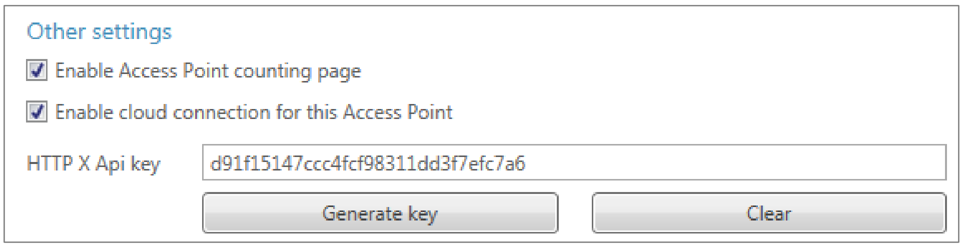
Besides JSON format, it is also possible to retrieve the data in XML format. In case XML output is required, the "**'Accept': 'application/xml'**" header must be sent with the HTTP request.

When the HTTP request is successful, the collected counting data is returned in XML format.

4.1.2 Authorization

To avoid unwanted data requests, the security level can be increased by activating an authorization key on the CrossCONNECT Access Point.

Use Device Explorer to navigate to the IP settings page of the CrossCONNECT Access Point and click the **Generate key** button. This will generate a random key.



Other settings

☑ Enable Access Point counting page

☑ Enable cloud connection for this Access Point

HTTP X Api key: d91f15147ccc4fcf98311dd3f7efc7a6

Figure 23: Generating an X-Api-Key

When this key is generated, the header "**'X-Api-Key': 'thevalueofyourkey'**" must be sent as a header in the HTTP request.

Only when the specified key matches with the key that has been set in the CrossCONNECT Access Point, the counting data will be returned. In case of a mismatch, the result code **401 Not Authorized** will be returned.

4.1.3 HTTP request result codes

The following HTTP result codes can be returned by the HTTP request:

- **200** (OK) – is returned for every successful request.
- **503** (Service Unavailable) – the service on the Access Point is inactive. Restart the Access Point to start the service again.
- **401** (Not Authorized) – The required X-Api-Key is not specified or does not match the key which is set on the CrossCONNECT Access Point.
- **404** (Not Found) – a wrong command was sent to the Access Point.
- **429** (Too Many Requests) – the maximum number of allowed requests within the time-frame specified below under Limitations has been reached.

4.1.4 Data result values

The "result" property inside the actual data response can have one of the following values:

- **ok** (the counting data was successfully collected)
- **invalid-arguments** (a non-existing argument and/or a wrong value of an argument is detected)
- **fieldbus-error** (errors on fieldbus level or disconnected devices are detected)

4.1.5 Data contents

The data response contains information on the date, day of the week, counting totals sum of all entrances) and entrance specific information like entrance name and counting totals (if applicable for the selected counting type).

4.1.6 Limitations

The number of requests is limited to ten (10) requests per five (5) seconds. In case more requests are sent within this time-frame, an HTTP error with result code 429 (Too many Requests) will be returned.