Nagravision S.A.
IMS
Data Importer (Schedule File Interface 4X1)
Specifications
V 1.0.0

CH - 1033 CHESEAUX, SWITZERLAND ✆+41.21.732.03.11 📠+41.21.732.03.00

Table Of Contents

List of figures

List of tables

1 Introduction

1.1 Purpose

This document defines the Nagravision Data Importer schedule file interface, that is, the expected format of the schedule information transmitted to the target system, using the XML standard [2]. This document is mainly intended to software developers and designers.

In addition to the XML file format specification, this document also contains the functional specification of the importation interface.

1.2 Overview

This document describes the expected format of the XML files containing scheduling information. These files are processed by the Data Importer application to populate the target database.

XML has been chosen to formalize the data definition for the following reasons:

- It is a widely accepted standard.
- It allows extensions without impacting previous implementations (backward compatibility).
- The format specification is directly usable by the application (no inconsistencies).
- The specification can be customized (to some extent) without changing the software.
- The format is easily readable without any particular tool.
- Data can be converted to HTML and displayed by a standard Internet browser.

1.3 Notational Conventions

- A new concept, component, function etc. that is introduced for the first time in the text is defined in the corresponding section and the corresponding term appears in *italic* type.
- XML specifications and other formal specifications appear in `Courier` font.

1.4 Definitions, Acronyms, and Abbreviations

Acronym Abbreviation	Definition
CAS	Conditional Access System
DTD	Document Type Definition
DVB	Digital Video Broadcasting. (www.dvb.org)
EPG	Electronic Program Guide
IMS	Information Management System (www.nagra.com)
ISO	International Organization for Standardization. (www.iso.ch)
XML	Extensible Markup Language (www.w3.org/xml)

1.5 References

[1] DVB, Specifications for Service Information (SI) in DVB systems, EN 300 468 v1.3.1, 1998-02

[2] W3C, Extensible Markup Language (XML) 1.0, REC-xml-19980210

2 Software Interface overview

2.1 System Interface

The Data Importer accepts import files from one or several *FTP file servers*, hosting files provided by one or several *Providers*.

The importer polls regularly each file server, processes potential import files and loads them into the target database instance.

The importer may also enforce some validation rules and verify that providers have the appropriate rights to perform the actions contained in the import file.

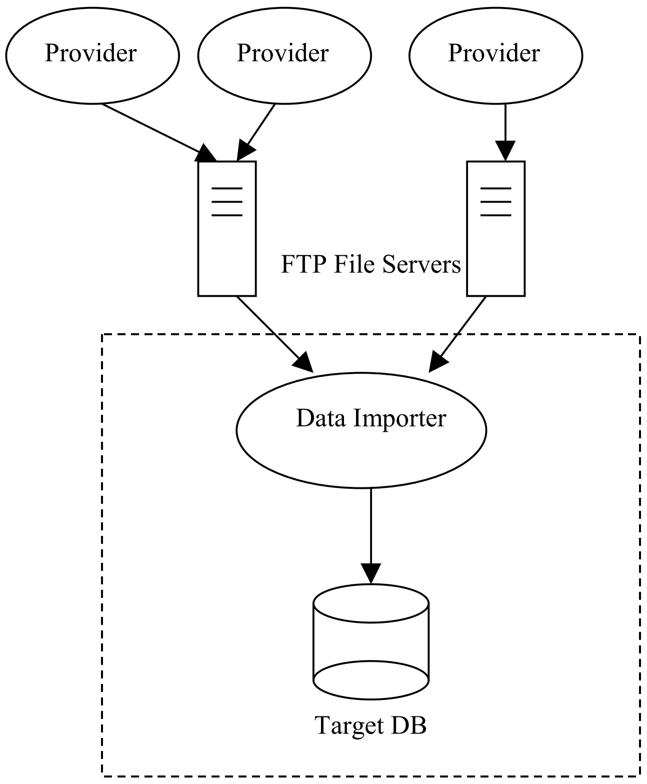

Fig. 2-1 System Interface

3 Data Grammar Specification

For convenience reasons, the grammar specification is provided in appendix files.

The HTML doc should facilitate the readability of the grammar on a browser, while the XML Schema (or, optionally, the DTD) is the formal XML specification to be used for development & test:

- BroadcastDataDoc.html (HTML on-line documentation)
- BroadcastData.xsd (XML-Schema)
- BroadcastData.dtd (Document Type Definition – DTD)

Note: The HTML doc, DTD and XML Schema have all been automatically generated from a same specification source. Therefore any inconsistencies between those different views are highly unexpected.

4 Data Importer Functional Specification

4.1 File Handling

4.1.1 File Provision

As shown in section 2.1, Providers use a specific FTP File Server to upload import files. The Importer accesses this file server using a dedicated user account and directory to fetch the files.

Import files are transferred between the server and the Importer host using FTP protocol as well.

When no remote file server is required or needed, import files may be directly stored on the localhost (where the Importer is running). A valid localhost user account is however still needed in that case for the Importer to fetch the files locally.

In both cases (local or remote), a dedicated directory *per Provider* must be configured to hold import files to process.

The following figure shows the subdirectory structure required by the Importer and the operational steps of file processing:

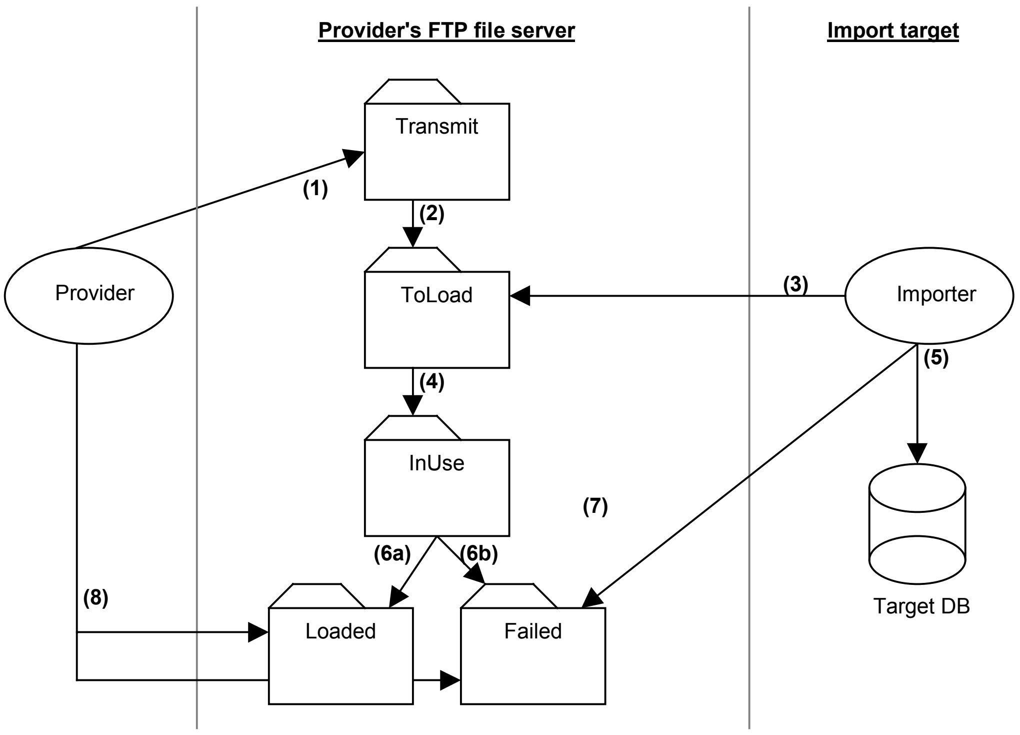

Fig. 4-1: Schedule File handling

1) The *Provider* uploads new import files in the *Transmit* subdirectory of his dedicated server's account.

2) At file transfer's completion (which may take a while if the file is big and the connection bitrate low!), the *Provider* moves the import file into the *ToLoad* directory (using *FTP rename*

command, which is instantaneous). That way, we ensure the *Importer* won't see a file before it has been fully transferred.

3) The *Importer* periodically polls the server for new files to import. If a new file is found, the file is FTP-transferred on the local host for processing.

4) The *Importer* also moves the file from the *ToLoad* directory to the *InUse* directory, to indicate that it is being processed.

5) The file is processed and the target database is updated accordingly.

6) After the file has been processed, the *Importer* moves it to the *Loaded* directory if no error occurred (6a) or to the *Failed* directory if error(s) occurred (6b).

7) In case of failure (6b), the errorlog file (see section 4.5) is also copied into the *Failed* directory.

8) The *Provider* may poll the *Loaded/Failed* directories to ensure his file was successfully processed. If the processing failed, it may fetch the errorlog file to analyze the problem and possibly re-submit a corrected file later on (see section 4.5.3).

Note: the *Importer* does not manage the purge of processed file, for the good reason that it doesn't know if the *Provider* (or someone else) still needs to access them.

4.1.2 File Naming Convention

The import files must follow the following naming convention (the date field contains the file creation date in GMT time):

```
provider_prefix + "_" + YYYYMMDDHHmmSS + ".xml"

Example: xyz_20000721131005.xml
```

To report errors, the Importer will reuse the exact import filename, with an additional "`.errorlog`" extension. The errorlog files will be stored in the *Failed* directory (see previous section).

4.1.3 File Compression

To reduce file transfer time and cost, import files may be compressed.

Due to the repetitive structure of XML, compression is actually strongly advised since it offers very good compression ratio.

The following table compares results obtained with various tools. Note that the tests were performed on a file generated automatically, therefore even more repetitive than a normal XML file. The results may thus be too optimistic, but should still give a good idea.

```
XML source file size: 9769962 bytes
```

Compression Tool	*Compressed Size*	*Compression Ratio*	*Compression Time*	*Decompression Time*
bzip2 (.bz2)	149293 bytes	65.4x	45 secs.	7 secs.
gzip (.gz)	377022 bytes	25.9x	6 secs.	3 secs.
compress (.Z)	720477 bytes	13.6x	5 secs.	4 secs.

Table 4-1: Compression tools comparison

The correct filename extension (`.bz2, .gz, .Z`) <u>must</u> be used in that case, since it's the only way for the importer to determine the decompression tool to use.

```
Example: xyz_20000721131005.xml.gz
```

The Data Importer will then decompress the file automatically, using the appropriate tool.

<u>Note:</u> Support of bzip2 is subject to tool's availability on the Data Importer deployment platform.

4.1.4 Delayed File Processing

The Provider may wish to "schedule" the loading of an import file at a specific future date and time.

This is possible using an additional filename extension. This extension must be:

```
".load_at_" + YYYYMMDDHHmmSS

Example: xyz_20000721131005.load_at_20000724060000.xml
```

When the Importer detects files containing that extension, it will delay the importation until the given loading date and time (GMT).

Note that this mechanism is not appropriate for real-time commands (e.g., the effective loading time of the file - and thus the activation of the change - might occur slightly later than the given time.)

Moreover, since the file is not validated beforehand, there is no guarantee that the change will succeed at all. For these reasons, this mechanism should be used with care, and only for changes requiring "minute" precision rather than "second" precision.

4.2 Insertion Rules

4.2.1 Schedule Data

The Schedule Data section is divided in *Channel Periods*. A Channel Period completely defines the schedule for the given channel and the given period of time. It contains the set of all events appearing on the channel during that period, *chronologically ordered* (e.g. sorted by begin time ascending).

The Schedule Data section also supports Productions. A production may be defined when a same event is scheduled multiple times (typically, in NVoD context). A globally unique alphanumeric ID must identify the production. In that case, events scheduled in a channel period can refer to an associated production, instead of specifying each time the full event details (see examples in section 4.5.3). Definition of Productions and Channel Periods are independent. There is no need to redefine the Production each time a Channel Period referring to it is imported. The only requirement is to define the Production at least once before referring to it in a schedule.

The ChannelPeriod must be wide enough to accommodate all contained events. Note that the channel period definition is used to "clear" the previous schedule during that period. So, if the contained events would not entirely cover the channel period, there would be gaps (holes) at the extremities of the period, which is normally not allowed. Similarly, time gaps are principally not allowed between consecutive events. Events' overlaps are of course not allowed (see also section 4.3).

The discrimination between event insertion and event update is based on the unique id of the element (e.g. EventId). If an event with a given EventId already exists in the database, the element is updated, otherwise it is considered new and is therefore inserted.

The explicit deletion of an event is not supported. However, when a new channel period is loaded, existing events in the period are deleted (unless they are present in the new schedule as well, in which case they are simply updated). A noticeable exception concerns events already sold in a product. These won't be deleted, but moved to a special channel instead, the Placeholder channel. Later on, it will be possible to reactivate them when the new scheduling is known.

The following figure illustrates the Channel Period insertion rules, and the relation between *imported schedule*, *current DB schedule* and *resulting DB schedule.*

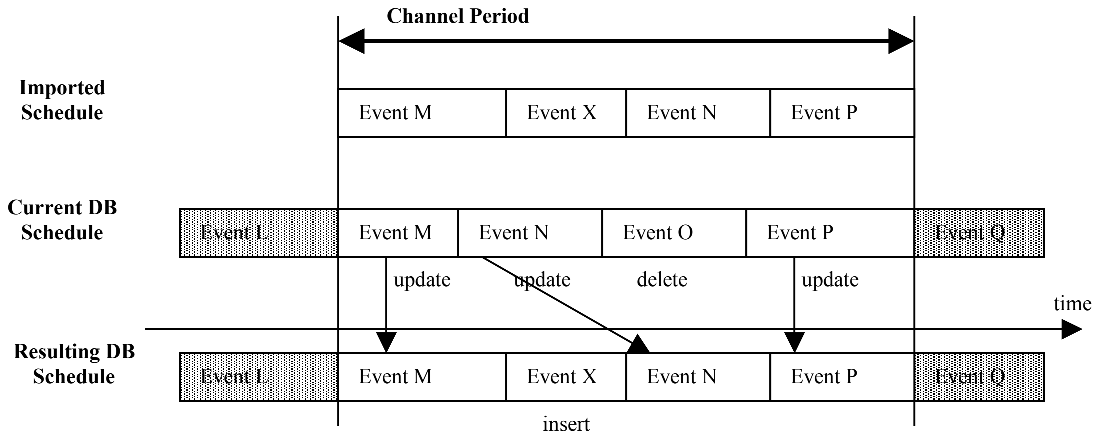

Fig. 4-2: Channel Period definition

Note that the boundaries of an imported Channel Period (e.g. period start time & end time) must not conflict with an existing schedule in the database.

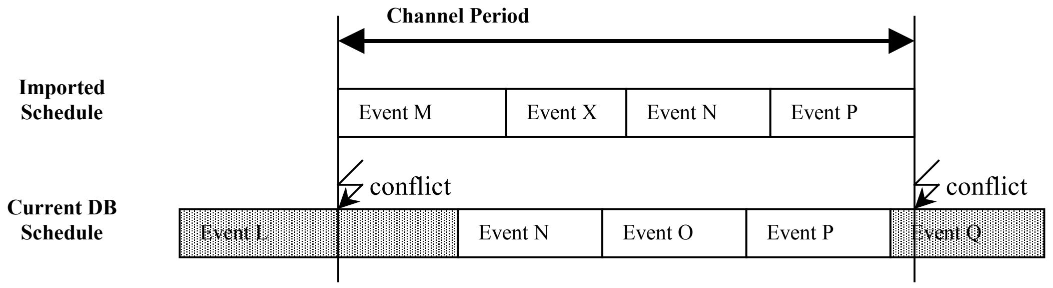

Fig. 4-3: Channel Period boundaries conflict

If such a situation occurs, the whole Channel Period is rejected, since the Data Importer does not know how to handle the events overlapping the period boundaries (Event L and Q in the example).

ChannelPeriod blocks are defined in total independence from one and another in the same ScheduleData block. Although this would serve no useful purpose, two ChannelPeriods applying to the same channel and period could be defined in the same file; of course, the resulting database would match the ChannelPeriod block content appearing last in the file.

In a more useful way, this behavior may be used to transmit only changes to an existing schedule. If the only changes are a few modified event descriptions, for example, one needs not to transmit the entire schedule again. It is sufficient to create a file containing as many ChannelPeriods as changed events, each enclosing the event in their period, and to include in each block only the changed event.

The Data Importer uses the channel period block as a natural transaction delimiter. Any error within a channel period block leads to the rejection of the entire block. The processing continues with the following block, if any.

4.3 Validation Rules

The Data Importer may apply multiple business-oriented validation rules, acting like a chain of filters on imported data.

The most common validation rules are described in this section. Other rules might exist according to additional customer requirements.

4.3.1 New Schedule

This validation rule, if active, prevents loading any schedule file with a creation date older than the last loaded file. This ensures that obsolete schedule files are not mistakenly (re)loaded, thus erasing more recent schedules.

Unlike other validation rules, this rule applies at file level, rather than at segment level, meaning that the file will be globally accepted or rejected.

4.3.2 No Overlap

This validation rule, if active (which should always be the case), ensures that the imported file does not contain any event overlaps (e.g. events starting before the end of the preceding one), or sub-event overlaps within an event (when micro-scheduling applies).

In addition, the rule may also check for the presence of gaps in the schedule. Generally, schedule gaps are not well tolerated by STBs, and it is preferable to use dummy filler events for off-air periods, rather than leaving the schedule undefined. However, rejection or acceptance of schedule containing gaps is a configurable option of the rule.

4.3.3 In Future

This validation rule, if active, ensures that the imported file does not alter the schedule too close before broadcast time (late changes).

The limit alteration date is given by:

```
    minAlterationDate = currentDate + noUpdateDelay
```

`noUpdateDelay` is a configurable parameter, ranging typically from 30 minutes to several hours, depending on the site requirements.

In principle, each event in the channel period is validated against this rule, and the beginning of the imported period is automatically aligned with the beginning of the first acceptable event.

Alternatively, the validation rule may also be configured so that non-compliant channel periods (e.g. containing at least one event starting before the limit date) are fully rejected.

4.3.4 Event ID

This validation rule, if active (which should always be the case), ensures that imported PPV (or potential PPV) events do have an Event ID.

Note that Event ID is not required for events on subscription channels, but it is allowed (and sometimes preferred).

4.4 Segments & Transactions

To cope with large files, we divide them into smaller processing segments. Import file segments are implicit and based upon specific XML elements token. In the present specification scope, the following segmentation elements apply:

- Production
- ChannelPeriod

Since data are imported into a database, we also have to define transactional blocks. Currently, each file segment corresponds to a DB transaction (e.g. globally committed or rolled-back).

This 1-to-1 relation between file segment and DB transaction has been chosen for simplicity and convenience reasons. We could envisage, for instance, to group several segments into a transaction. This would be useful if some segments are inter-dependents, and should not be committed individually.

4.5 Error Handling

Any error detected during processing of an import file are logged into an errorlog file. Since files are processed per segments (see 4.4), error logging is also structured per segment.

The processing of each segment is divided into phases. Those phases are:

1) Parsing
2) Formatting
3) Validation
4) Insertion

In each phase, errors may be detected. If an error occurs in one of the above phases, the processing phase will be completed (in order to detect all errors relevant for this phase), the current segment processing will be aborted, and the processing will go on with the next segment.

If one or more errors have occurred in a segment, the DB transaction associated to the segment will be entirely rolled-back.

If one or more segments have failed, the whole file will be notified as *Failed*. However, if the import file contains several segments, failure of one segment does not preclude further processing of the file and successful committing of other DB transactions. Therefore, a *Failed* file does not necessarily indicate that the whole file content has been ignored and rejected.

4.5.1 Errorlog format

The errorlog file is an XML document. It is structured as follow:

<u>ErrorLog:</u> Top-level element of the errorlog document.

<u>Segment:</u> Invalid segment error information. Each segment has two attributes; an id (name of segment's top-level element), and the segment's start line in the XML import source document.

ErrorInfo: Segment individual errors. The first attribute indicates the processing phase. The second attribute is an error code (currently unused and always defaulted to –1). A third optional attribute indicates the error line in the source document (parsing phase only). The element's body contains the error description.

Example:

```
<ErrorLog>
  <Segment id="ChannelPeriod" line="8">
    <ErrorInfo code="-1" line="9" phase="Parsing">
      Unknown element ‘DUMMY’
    </ErrorInfo>
    <ErrorInfo code="-1" line="346" phase="Parsing">
      Element ‘DUMMY’ is not valid for content model: ‘(ChannelId,Event*)’
    </ErrorInfo>
  </Segment>
</ErrorLog>
```

4.5.2 Errorlog sample

In this section, we examine how faulty elements of an example source schedule file (4.5.2.1) are detected and reported in the errorlog file (4.5.2.2). Each error is then commented and explained in details (4.5.2.3).

4.5.2.1 Source schedule file

```
 1. <?xml version="1.0" encoding="UTF-8"?>
 2. <BroadcastData creationDate="20020325093719">
 3.   <ProviderInfo>
 4.     <ProviderId>Nagra</ProviderId>
 5.     <ProviderName>NagraVision S.A.</ProviderName>
 6.   </ProviderInfo>
 7.   <ScheduleData>
 8.     <ChannelPeriod beginTime="20020325060000" endTime="20020325120000">
 9.       <ChannelId>100</ChannelId>
10.       <Event beginTime="20020325060000">
11.         <EventId>4000</EventId>
12.         <EventType>P</EventType>
13.         <EpgProduction>
14.           <EpgText language="english">
15.             <Name>Name of event 4000 (eng)</Name>
16.           </EpgText>
17.         </EpgProduction>
18.       </Event>
19.       <Event beginTime="20020325080000" duration="7200">
20.         <EventId>4001</EventId>
21.         <EventType>Z</EventType>
22.         <EpgProduction>
23.           <EpgText language="eng">
24.             <Name>Name of event 4001 (eng)</Name>
25.           </EpgText>
26.         </EpgProduction>
27.       </Event>
28.     </ChannelPeriod>
29.     <ChannelPeriod beginTime="20020325060000" endTime="20020325120000">
30.       <ChannelId>101</ChannelId>
31.       <Event beginTime="20020325060000" duration="7200">
32.         <EventId>4003</EventId>
33.         <EventType>P</EventType>
34.         <EpgProduction>
35.           <EpgText language="eng">
36.             <Name>Name of event 4003 (eng)</Name>
37.           </EpgText>
38.         </EpgProduction>
39.       </Event>
40.       <Event beginTime="20020325075000" duration="7200">
41.         <EventId>4004</EventId>
42.         <EventType>P</EventType>
43.         <EpgProduction>
```

```
44.           <EpgText language="eng">
45.             <Name>Name of event 4004 (eng)</Name>
46.           </EpgText>
47.         </EpgProduction>
48.       </Event>
49.     </ChannelPeriod>
50.     <ChannelPeriod beginTime="20020325060000" endTime="20020325080000">
51.       <ChannelId>ChannelXYZ</ChannelId>
52.       <Event beginTime="20020325060000" duration="7200">
53.         <EventId>4006</EventId>
54.         <EventType>P</EventType>
55.         <EpgProduction>
56.           <EpgText language="eng">
57.             <Name>Name of event 4006 (eng)</Name>
58.           </EpgText>
59.         </EpgProduction>
60.       </Event>
61.     </ChannelPeriod>
62.   </ScheduleData>
63. </BroadcastData>
```

4.5.2.2 Resulting errorlog file

```
<ErrorLog>
  <Segment id="ChannelPeriod" line="8">
    <ErrorInfo code="-1" line="10" phase="Parsing">
      Required attribute ‘duration’ was not provided
    </ErrorInfo>
    <ErrorInfo code="-1" line="14" phase="Parsing">
      Datatype error: Type:InvalidDatatypeValueException,
      Message:Value ‘english’ with length ‘7’ exceeds maximum length facet of ‘3’.
    </ErrorInfo>
    <ErrorInfo code="-1" line="21" phase="Parsing">
      Datatype error: Type:InvalidDatatypeValueException,
      Message:Value ‘Z’ is not in enumeration .
    </ErrorInfo>
  </Segment>
  <Segment id="ChannelPeriod" line="29">
    <ErrorInfo code="-1" phase="Validation">
      Event 4004 at 25 Mar 2002 07:50:00 is overlapped by previous event!
    </ErrorInfo>
    <ErrorInfo code="-1" phase="Validation">
      Channel Period end at 25 Mar 2002 12:00:00 leaves a gap after last event!
    </ErrorInfo>
  </Segment>
  <Segment id="ChannelPeriod" line="50">
    <ErrorInfo code="-1" phase="Insertion">
      Element corresponding to ‘<CHANNEL_PERIOD END_TIME=20020325080000
      BEGIN_TIME=20020325060000 CUST_CHANNEL_ID=ChannelXYZ/’> is not the DB,
      thus breaking the preexistence requirement!
    </ErrorInfo>
  </Segment>
</ErrorLog>
```

4.5.2.3 Comments

The source file contained 3 channel periods. Each channel period contained errors related to a specific phase of the processing.

The first channel period contained syntactic and data types errors, which are typically detected in the *parsing* phase. Those errors were:

1. The mandatory duration attribute of Event was missing (1st event)
2. The language attribute of EpgText was too long, with 7 chars instead of 3 maximum according to XML Schema (1st event)
3. The EventType ‘Z’ is not allowed according to XML Schema (2nd event)

We can note that parsing errors are reported with a reference to the exact line number were the error was detected in the source document.

The second channel period contained business-logic errors, which are typically detected in the *validation* phase. Those errors were:

1. The 2nd event starts 10 minutes before the end of the previous event, implying an obvious event overlap situation.
2. The last event ends earlier than the global channel period does, leaving a schedule gap of 2 hours 10 minutes between its end and the end of the channel period. Assuming that schedule gap validation constraint is enabled (no gaps allowed), this is not correct.

We can note that validation errors are not reported with a reference to a line number in the source document. This is because business-logic errors can generally not be related unequivocally to a precise source document line. However, validation error messages contain normally enough information to clearly identify the error source (e.g. event id & event start in that case).

The third channel period contained DB-related errors, which are typically detected in the *insertion* phase. The only error illustrated was:

1. No Channel matching the given Channel Id was found in the DB, thus preventing the possibility to insert any schedule on a non-existing channel.

Fortunately, most of the errors are detected before the insertion phase. Insertion phase errors are generally the most difficult to understand and to correct, for several reasons:

- Since bulk-insertion mode is used, the exact row causing the error cannot be identified, and, a fortiori, the source document errors line either.
- Direct DB error messages are often cryptic and not easily understandable.
- Severe inconsistencies between the data provision source and the target system generally only show up in that phase.

Most common insertion phase errors are:

1. An object referred by the source file is not present in the DB (like the ChannelXYZ in the example).
2. There is a conflict between imported data and current DB data (for instance, the imported channel period boundaries overlap some existing events).
3. The DB data model differs from the XML-Schema model (normally never happens in a properly tested & integrated system).

4.5.3 Failed files handling

In order to react to errors, the Provider should be able to:

a) Detect the problem.
b) Correct the problem.
c) Resubmit the corrected file / file segment.

The above induces the following requirements for the Provider:

a) To be able to check the processing result of a submitted file.
b) To be able to notify appropriate personnel in case of failure (automatic correction is not a reasonable option).
c) To be able to re-generate error segments of a failed import file. (=> Some tool to trigger the generation process).

A simplified approach is to correct failures directly into the import file (possibly without even notifying the provider).

5 Samples

5.1 ChannelPeriod import file (typical subscription)

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadcastData creationDate="20000721160000">
  <ProviderInfo>
    <ProviderId>Nagra</ProviderId>
    <ProviderName>NagraVision S.A.</ProviderName>
  </ProviderInfo>
  <ScheduleData>
    <ChannelPeriod beginTime="20001209060000" endTime="20001210060000">
      <ChannelId>BBC1</ChannelId>
      <Event beginTime="20001209060000" duration="7200">
        <EventType>S</EventType>
        <EpgProduction>
          <EpgText language="eng">
            <Name>Titanic</Name>
            <ShortDescription>The dramatic story of Titanic</ShortDescription>
            <Description>A ship hits an iceberg on his way to NY</Description>
            <ExtendedInfo name="Actors">L. Di Caprio, K. Winslet</ExtendedInfo>
            <ExtendedInfo name="Director">James Cameron</ExtendedInfo>
          </EpgText>
          <ProtectionMode>1</ProtectionMode>
          <ParentalRating>9</ParentalRating>
          <AudioInfo>
            <Stereo>1</Stereo>
            <Dolby>2</Dolby>
            <Surround>1</Surround>
          </AudioInfo>
          <VideoInfo>
            <WideScreen>1</WideScreen>
          </VideoInfo>
          <DvbContent>
            <Content nibble1="1" nibble2="0"/>
            <User nibble1="3" nibble2="a"/>
          </DvbContent>
          <UrlInfo>http://www.titanic.com</UrlInfo>
        </EpgProduction>
      </Event>
      <Event beginTime="20001209080000" duration="5400">
        <EventType>S</EventType>
        <PrivateDescriptor tag="62" length="5">0AC2AB132E</PrivateDescriptor>
        <EpgProduction>
          <EpgText language="eng">
            <Name>Terminator</Name>
            <Description>Arnie goes back into the past</Description>
          </EpgText>
          <ParentalRating>5</ParentalRating>
        </EpgProduction>
      </Event>
      <Event beginTime="20001209093000" duration="5400">
        <EventType>S</EventType>
        <EpgProduction>
          <EpgText language="fra">
            <Name>Le Grand Bleu</Name>
            <Description>Histoire d'amour entre un plongeur et un dauphin</Description>
          </EpgText>
          <ParentalRating>6</ParentalRating>
        </EpgProduction>
      </Event>
    </ChannelPeriod>
  </ScheduleData>
</BroadcastData>
```

5.2 ChannelPeriod import file (typical NVoD)

```
<?xml version="1.0" encoding="UTF-8"?>
<BroadcastData creationDate="20000721160000">
  <ProviderInfo>
    <ProviderId>Nagra</ProviderId>
    <ProviderName>NagraVision S.A.</ProviderName>
  </ProviderInfo>
  <ScheduleData>
    <Production>
      <ProductionId>FR123000</ProductionId>
      <ProductionTitle>Titanic</ProductionTitle>
      <EpgProduction>
        <EpgText language="eng">
          <Name>Titanic</Name>
          <Description>A ship hits an iceberg on his way to NY</Description>
          <ExtendedInfo name="Actors">Leonardo Di Caprio, Kate Winslet</ExtendedInfo>
          <ExtendedInfo name="Director">James Cameron</ExtendedInfo>
        </EpgText>
        <ParentalRating>9</ParentalRating>
        <DvbContent>
          <Content nibble1="1" nibble2="0"/>
          <User nibble1="3" nibble2="a"/>
        </DvbContent>
        <UrlInfo>http://www.titanic.com</UrlInfo>
      </EpgProduction>
    </Production>
    <ChannelPeriod beginTime="20001209060000" endTime="20001210060000">
      <ChannelId>NVOD1</ChannelId>
      <Event beginTime="20001209060000" duration="7200">
        <EventId>123000</EventId>
        <EventType>P</EventType>
        <ProductionId>FR123000</ProductionId>
      </Event>
      <Event beginTime="20001209080000" duration="7200">
        <EventId>123001</EventId>
        <EventType>P</EventType>
        <ProductionId>FR123000</ProductionId>
      </Event>
      <Event beginTime="20001209100000" duration="7200">
        <EventId>123002</EventId>
        <EventType>P</EventType>
        <ProductionId>FR123000</ProductionId>
      </Event>
    </ChannelPeriod>
    <ChannelPeriod beginTime="20001209063000" endTime="20001210063000">
      <ChannelId>NVOD2</ChannelId>
      <Event beginTime="20001209063000" duration="7200">
        <EventId>123010</EventId>
        <EventType>P</EventType>
        <ProductionId>FR123000</ProductionId>
      </Event>
      <Event beginTime="20001209083000" duration="7200">
        <EventId>123011</EventId>
        <EventType>P</EventType>
        <ProductionId>FR123000</ProductionId>
      </Event>
      <Event beginTime="20001209103000" duration="7200">
        <EventId>123012</EventId>
        <EventType>P</EventType>
        <ProductionId>FR123000</ProductionId>
      </Event>
    </ChannelPeriod>
  </ScheduleData>
</BroadcastData>
```

—— End of document ——