3GPP ASN.1 Gateway User Guide

Release: 3.4.0

Date: February 5, 2008

Contents

References

Name	Description
[Gateway Framework User Guide]	This use guide describes in detail the functionality of the Gateway Framework, and the standard suite of tools available.
[3GPP TS 32.401]	The 3GPP TS 32.401 document is part of the 32.400-series covering the 3rd Generation Partnership Project technical specification of Telecom performance management requirements. The document describes the requirements for the management of performance measurements and the collection of performance measurement result data across GSM and UMTS networks.
[3GPP TS 32.403]	Describes the mechanisms involved in the collection of the data and the definition of the UMTS and combined UMTS/GSM performance data itself.

Glossary

3GPP	3rd Generation Partnership Project
ASN.1	Abstract Syntax Notation 1
MGW	Media Gateway
FTP	File Transfer Protocol
GSM	Global System for Mobile communications
GPRS	General Packet Radio Service
NE	Network Element
NM	Network Manager
NRM	Network Resource Models.
OA&M	Operation, Administration and Maintenance
TS	Technical Specification
UMTS	Universal Mobile Telecommunications System
UTRAN	Universal Terrestrial Radio Access Network
XML	EXtended Markup Language

Preface

About this Guide

This guide details the vendor specific information on the 3GPP ASN.1 Gateway. It contains the following information:

- *Chapter 1. Overview.* This chapter gives a brief description of the 3GPP ASN.1 Gateway and the raw data format it parses.
- *Chapter 2: Engine Rules and Configuration.* This chapter details the rules and their configuration for parsing vendor specific raw data.
- *Chapter 3: Post Parser Rules and Configuration.* This chapter describes Post Parser rules and their configuration provided with this parser.

Conventions

The following conventions are used in this guide:

`Fixed width`	Highlights a block of example code, a configuration entry, or a command line instruction

1. Overview

1.1 The Gateway Framework

The 3GPP ASN.1 Gateway uses the Gateway Framework as a container for the execution of its engine and post parser stages. The Gateway Framework and the Gateway are decoupled into two separate installations. The Gateway Framework consists of a library of Perl modules that provide functionality such as:

- a container for the execution of the Engine and Post Parser rules for data transformation
- Intermediate (PIF) and output data (LIF) storage and management
- logging utilities
- cleanup and crash recovery
- statistics gathering

The Gateway plugs into the Gateway Framework and extends this functionality to provide the final Gateway that processes the vendor specific data. More information on the standard Gateway configuration is contained in the [Gateway Framework User Guide].

1.2 3GPP ASN.1 Gateway Overview

The following is an overview of the Gateway.

1.2.1 3GPP

3GPP stands for 3rd Generation Partnership Project. 3GPP is a collaboration agreement that brings together a number of telecommunications standards bodies, which are known as organizational partners. The main purpose of the partnership is to produce technical specifications and reports for the 3rd Generation mobile system.

One of these specifications, the [3GPP TS 32.401] covers the requirements for the collection and management of performance measurements result data across GSM and UMTS networks. It defines the administration of measurement schedules by the Network Element Manager (EM), the generation of measurement results in the Network Elements (NEs) and the transfer of these results to one or more systems. An example of such systems, north-bound from the data source are the Vallent performance management products.

Another specification of interest is the [3GPP TS 32.615] which defines the configuration management XML file format definition required for defining Network Resource Models (NRM).

1.2.2 ASN.1

Abstract Syntax Notation One (more commonly known as ASN.1) is a language for defining standards without regard to the implementation. It is the language of standards writers. For example, if a standards writer submitting to a standards body wants to write a recommendation for standardizing the procedures that one component follows for talking to another component, he writes the recommendation in ASN.1 notation, and submits the recommendation to a standards body such as the ITU. ASN.1 facilitates communication between professionals and committee members by offering a common language for describing a standard. ASN.1 is defined in ITU-T Recommendations X.209 and X.690.

For example, ASN.1 defines:

- What a "type" is.
- What a module is and how it should look.
- What a BOOLEAN is.
- What a "structured type" is.
- What certain keywords mean (for example, BEGIN, END, IMPORT, EXPORT, EXTERNAL, and so on).
- How to "tag" a type, so that it can be properly encoded.

ASN.1 has no regard to any specific standard, encoding method, programming language, or hardware platform. It is simply a language for defining standards. Or in other words, standards are written in ASN.1.

1.2.2.1 Encoding Rule for ASN.1 - BER

BER is the common name for the Basic Encoding Rules of ASN.1. BER is defined in ITU-T Recommendations X.209 and X.690. BER is one set of rules for encoding ASN.1 data to a stream of octets that can be

transmitted over a communications link. Other methods of encoding ASN.1 data include Distinguished Encoding Rules (DER), Canonical Encoding Rules (CER), and Packing Encoding Rules (PER). Each encoding method has its application, but BER tends to be the encoding method most commonly used and most commonly talked about.

BER defines:

- Methods for encoding ASN.1 values.
- Rules for deciding when to use a given method.
- The format of specific octets in the data.

1.2.2.2 Contrasting ASN.1 and BER

ASN.1 is like a programming language (such as C), whereas BER is like a compiler for that language. Compilers are platform-specific, whereas many high-level programming languages are not. C defines the rules and language for writing a program. The program is not useful until it is compiled for a specific platform (such as Intel x86). So it is with ASN.1 and BER. ASN.1 is the language for writing a standard. A standard is not ASN.1; it is written in ASN.1. Data which is generated from a program, that complies with the standard may loosely be termed "ASN.1 data." ASN.1 data is not useful (that is, it cannot be transmitted across a LAN) until the ASN.1 data is encoded into a stream of octets, which can be easily decoded at the destination.

1.2.3 Network Details

Several of the vendors who supply GPRS and UMTS equipment to Vallent customers produce performance statistics and network configuration management data from their equipment and OA&M management systems that complies with the 3GPP specifications in ASN.1 or XML format.

1.2.4 Data/File Formats

The [3GPP TS 32.401] standard supports both the ASN.1 and XML data format. Here we are only concerned with the ASN.1 option.

The following outlines the data and filename formats processed by the Gateway.

1.2.4.1 Performance Data Layout

Whether in ASN.1 or XML format, the data is organised in the data structure definition described in table 1.

ASN.1 Tag	DTD based XML Tag	XML schema based XML tag	Description
MeasDataCollection	Mdc	MeasCollecFile	This is the top-level tag, which identifies the file as a collection of measurement data. The file content is made up of a header ("measFileHeader"), the collection of measurement result items ("measData"), and a measurement file footer ("measFileFooter").
MeasFileHeader	Mfh	FileHeader	This is the measurement result file header to be inserted in each file. It includes a version indicator, the name, type and vendor name of the sending network node, and a time stamp ("collectionBeginTime").
MeasData	Md	MeasData	The "measData" construct represents the sequence of zero or more measurement result items contained in the file. It can be empty in case no measurement data can be provided. Each "measData" element contains the name of the NE ("nEId") and the list of measurement results pertaining to that NE ("measInfo").
MeasFileFooter	Mff	FileFooter	The measurement result file footer to be inserted in each file. It includes a time stamp, which refers to the end of the overall measurement collection interval that is covered by the collected measurement results being stored in this file.
FileFormatVersion	Ffv	fileHeader fileFormatVersion	This parameter identifies the file format version applied by the sender. The format version defined in the present document shall be the abridged number and version of the 3GPP document for XML formats and the ASN.1 format alike.
SenderName	Sn	fileHeader dn Prefix and fileSender localDn	The senderName uniquely identifies the NE or EM that assembled this measurement file by its Distinguished Name (DN), according to the definitions in 3GPP TS 32.300 [10]. In the case of the NE-based approach, it is identical to the sender's "nEDistinguishedName". For ASN.1 and DTD based XML format, the string may be empty (i.e. string size =0) in case the DN is not configured in the sender.
SenderType	St	fileSender elementType	This is a user configurable identifier of the type of network node that generated the file, e.g. NodeB, EM, SGSN. The string may be empty (i.e. string size =0) in case the "senderType" is not configured in the sender.
VendorName	Vn	fileHeader vendorName	The "vendorName" identifies the vendor of the equipment that provided the measurement file.
CollectionBeginTime	Cbt	measCollec beginTime	The "collectionBeginTime" is a time stamp that refers to the start of the first measurement collection interval (granularity period) that is covered by the collected measurement results that are stored in this file.
NEId	Neid	ManagedElement	The unique identification of the NE in the system. It includes the user name ("nEUserName"), the distinguished name ("nEDistinguishedName") and the software version ("nESoftwareVersion") of the NE.
NEUserName	Neun	ManagedElement userLabel	This is the user definable name ("userLabel") defined for the NE. The string may be empty (i.e. string size =0) if the "nEUserName" is not configured in the CM applications.

ASN.1 Tag	DTD based XML Tag	XML schema based XML tag	Description
NEDistinguishedName	Nedn	fileHeader dnPrefix and managedElement localDn	This is the Distinguished Name (DN) defined for the NE. It is unique across an operator's 3G network. The string may be empty (i.e. string size =0) if the "nEDistinguishedName" is not configured in the CM applications.
NESoftwareVersion	Nesw	ManagedElement swVersion	This is the software version ("`swVersion`") defined for the NE. This is an optional parameter which allows post-processing systems to take care of vendor specific measurements modified between software versions.
MeasInfo	Mi	MeasInfo	The sequence of measurements, values and related information. It includes a list of measurement types ("measTypes") and the corresponding results ("measValues"), together with the time stamp ("measTimeStamp") and granularity period ("granularityPeriod") pertaining to these measurements.
MeasTimeStamp	Mts	GranPeriod endTime	Time stamp referring to the end of the granularity period.
JobId	Jobid	job jobId	The "jobId" represents the job with which measurement result contained in the file is associated. The "jobId" is mandatory when PMIRP is supported.
GranularityPeriod	Gp	granPeriod duration	Granularity period of the measurement(s) in seconds.
ReportingPeriod	Rp	repPeriod duration	Reporting period of the measurement(s) in seconds. The "reportingPeriod" is mandatory when PMIRP is supported.
MeasTypes	Mt	MeasTypes or measType	This is the list of measurement types (counter names) for which the following, analogous list of measurement values ("measValues") pertains.
MeasValues	Mv	MeasValue	This parameter contains the list of measurement results for the resource being measured, e.g. trunk, cell. It includes an identifier of the resource ("measObjInstId"), the list of measurement result values ("measResults") and a flag that indicates whether the data is reliable ("suspectFlag").
MeasObjInstId	Moid	measValue measObjLdn	The "measObjInstId" field contains the local distinguished name (LDN) of the measured object within the scope defined by the "nEDistinguishedName". The concatenation of the "nEDistinguishedName" and the "measObjInstId" yields the DN of the measured object. The "measObjInstId" is therefore empty if the "nEDistinguishedName" already specifies completely the DN of the measured object, which is the case for all measurements specified on NE level. For example, if the measured object is a "ManagedElement" representing RNC "RNC-Gbg-1", then the "nEDistinguishedName" will be for instance "DC=a1.companyNN.com,SubNetwork=1,IRPAgent=1,SubNetwork=CountryNN,MeContext=MEC-Gbg-1,ManagedElement=RNC-Gbg-1", and the "measObjInstId" will be empty. On the other hand, if the measured object is a "UtranCell" representing cell "Gbg-997" managed by that RNC, then the "nEDistinguishedName" will be for instance the same as above, i.e. "DC=a1.companyNN.com,SubNetwork=1,IRPAgent=1,SubNetwork=CountryNN,MeContext=MEC-Gbg-1,ManagedElement=RNC-Gbg-1", and the "measObjInstId" will be for instance "RncFunction=RF-1,UtranCell=Gbg-997".

ASN.1 Tag	DTD based XML Tag	XML schema based XML tag	Description
MeasResults	R	MeasResults or R	This parameter contains the sequence of result values for the observed measurement types. The "measResults" sequence shall have the same number of elements, which follow the same order as the measTypes sequence. Normal values are INTEGERs and REALs. The NULL value is reserved to indicate that the measurement item is not applicable or could not be retrieved for the object instance.
SuspectFlag	Sf	Suspect	Used as an indication of quality of the scanned data. FALSE in the case of reliable data, TRUE if not reliable. The default value is "FALSE", in case the suspect flag has its default value it may be omitted.
TimeStamp	Ts	MeasCollec endTime	ASN.1 GeneralizedTime format. The minimum required information within timestamp is year, month, day, hour, minute, and second.

Table 1 - Mapping of ASN.1 Measurement Report File Format tags to XML tags

1.2.4.2 File naming specification

A 3GPP compliant ASN.1 file will have to follow the naming convention as shown below:

```
<Type><Startdate>.<Starttime>-[<Enddate>.]<Endtime>_[<UniqueId>][_-
_<RC>]
```

- <Type> consist of (A/B/C/D)
  - A - Single NE, single granularity period.
  - B - Multiple NEs, single granularity period.
  - C - Single NE, multiple granularity periods.
  - D - Multiple NEs, multiple granularity periods.

- <Startdate> is the date when the granularity period begins (for A and B) or the first granularity period of the measurement result comes to availability (C and D). It takes the format of YYYYMMDD, where
  - YYYY - 4 digits year.
  - MM - 2 digits month (01-12).
  - DD - 2 digits day (01-31).

- <Starttime> is the time when the scenario mentioned above happens. It takes the format of HHMMshhmm, where
  - HH - 2 digits hour (00-23)
  - MM - 2 digits minute (00-59)
  - s - the sign for UTC time (+ or -)

- hh - 2 digits hour for local time differential from UTC time (00-23)
- mm - 2 digits minute for local time differential from UTC time (00-59)

The following are optional tokens in the filename:

- <Enddate> only exists if the type is C or D. It indicates the end of the last granularity period. The format is the same as <Startdate>.
- <Endtime> indicates when the granularity period (the last granularity period if type is C or D) ends. The format is same as <Starttime> but the minute portion is only in multiple of 5, i.e. (00, 05, 10, 15 etc.).
- <UniqueId> is the name of the NE, EM or domain. This field is optional.
- <RC> is a running count. It shall be appended only if the filename is not unanimous.

1.2.5 Architectural extensions

The Perl based 3GPP_ASN1 engine is the interface designed to convert from 3GPP compliant ASN.1 data to the Gateway PIF format.

The interface uses the Perl Convert::BER library to facilitate ASN.1 decoding and parsing. Convert::BER provides an OO interface to encoding and decoding data using the ASN.1 Basic Encoding Rules (BER), a platform independent way of encoding structured binary data together with the structure.

2. Engine Rules and Configuration

2.1 ASN1_3GPP

The essential task of the ASN1_3GPP engine is to parse the vendor specific 3GPP ASN.1-compliant data into output PIF files.

There will typically be a one-to-many relationship between one ASN.1 file, which will probably map to several PIF files. One PIF file will be produced for every measInfo ASN.1 tag element in the data, as this ASN.1 block represents the counter names and values for a particular network abstraction for a particular measurement period. Note that

the PIF file output will also include the header data common to the ASN.1 file.

A key goal of a successful configuration is to ensure that all the ASN.1 data is output to PIF files, and that no PIF duplication occurs. This duplication may occur if the configurations such as the PIF filename tokens do not give a unique filename for that particular measurement.

2.1.1 Convert::BER package

The ASN1_3GPP rule uses the Perl Convert::BER library to facilitate ASN.1 decoding and parsing. Convert::BER provides an OO interface to encoding and decoding data using the ASN.1 Basic Encoding Rules (BER), a platform independent way of encoding structured binary data together with the structure.

The Convert::BER::decode(opList) function decodes the data in the Convert::BER object buffer, using the opList data describing data structure. An opList is a list of operator-value pairs. An operator can be any of those defined below, or any defined by sub-classing Convert::BER, which will probably be derived from the primitives given here.

The values depend on whether BER is being encoded or decoded. For decoding, if the value is a reference to a scalar, decode the value into the scalar. If the value is a reference to a list, then decode all the items of this type into the list. Note that there must be at least one item to decode, otherwise the decoding will fail. If the value is a code reference, then execute the code and decode the value into the reference returned from the evaluated code.

2.1.2 Vendor specific Parser modules

The Gateway includes vendor specific parser modules to cater for vendor variations to the standard:

- ASN1_3GPP_BER_Ericsson.pm
- ASN1_3GPP_BER_Siemens_UTRAN.pm

A vendor specific parser module inherits from Convert::BER. It includes some tag definitions. Its key function is to invoke the Convert::BER::decode() function with the OpList designed on the 3GPP ASN.1 data structure description. The opList is embedded with Perl variables designed to extract data from the decoded structures during the decode invocation.

The net result of an invocation is a Perl hash data structure that contains all of the performance data in a structure designed to map easily to PIF output files.

2.1.3 Processing Sequence

The ASN1_3GPP processes the performance data in several steps as illustrated with the Ericsson parser module below:

1. Validate the performance data filename conforms to the 3GPP filename convention.
2. Instantiate the ASN1_3GPP_BER_Ericsson object.
3. Read the contents of the ASN.1 performance file into the ASN1_3GPP_BER_Ericsson object
4. Decode the ASN.1 buffer using the Convert::BER::decode() function. This function performs the BER decoding on the ASN.1 data, using the user configured ASN.1 data structures that describe the data be decoded.
5. The Convert::BER::decode() function returns a data structure designed to contain the PIF mappings from the ASN.1 data.
6. Iterate through the returned data structure and generate PIF file output for each PIF mapping.

2.1.4 Rule Configuration

The following details the vendor specific configuration rule entries for this Gateway.

- ASN1_HEADER_COUNTERS_FOR_DATE_TIME_MANIP: This array entry contains the list of counters from the parsed data, which require date and time related processing. The conversion will result in a value of 20050418091500 becoming two “date” and “time” counter values of 18Apr2004 and 15:00 respectively.

```
ASN1_HEADER_COUNTERS_FOR_DATE_TIME_MANIP => [
       qw(collectionBeginTime
          MeasTimeStamp
          MeasFileFooter
          StartTime
          StartDate) ],
```

- ASN1_ENABLE_PIF_COUNTER_IN_FILENAME: This is an optional scalar entry, when set to "True", enables the inclusion of a PIF counter entry in the PIF filename. This is required when all other

PIF tokens used to make up a PIF filename are identical over two or more PIF files to prevent PIF files from being overwritten. Sample configurations:

```
ASN1_ENABLE_PIF_COUNTER_IN_FILENAME  => 'True',
ASN1_ENABLE_PIF_COUNTER_IN_FILENAME  => 0
```

The following are some sample PIF output files containing counter values:

```
AtmInterface_80-#-04Oct2001_11:00:00-#-C-#-0-#-I.pif
AtmInterface_80-#-04Oct2001_11:00:00-#-C-#-1-#-I.pif
AtmInterface_80-#-04Oct2001_11:00:00-#-C-#-2-#-I.pif
```

- ASN1_ENABLE_PIF_PER_MOID: Normally the engine rule will create a PIF for each measInfo element. This can lead to a large number of difficult to manage files being created. If this option is set to true, the rule will create a PIF file for each measoObInstId key. So is there are a number of measInfo entries in the file for the same measObjInstId, they will output to the same PIF. If the counters associated with the measObjInstId change a new PIF is created.

```
ASN1_ENABLE_PIF_PER_MOID => 0,
```

- ASN1_3GPP_BER_PACKAGE_NAME: This mandatory scalar entry contains the name of the Perl module that is to be used to decode the ASN.1 3GPP based data into a data structure used to generate the output PIF's. The perl module, which will have a .pm extension, is actually a sub class of the Convert::BER 3rd party ASN.1 parsing package.

```
ASN1_3GPP_BER_PACKAGE_NAME  => 'ASN1_3GPP_BER_Ericsson',
```

- ASN1_OBJECT_INSTANCE_ID_ELEMENT_NAME: This scalar entry contains name of the element tag whose value contains the measurement Object Instance lD. It identifies the relative distinguished name of the measured object within the scope defined by the nEDistinguisedName. The concatenation of the nEDistinguisedName and the measObjInstld yields the DN of the measured object. This scalar entry allows for the ID of the current measured object to be included in each record.

```
ASN1_OBJECT_INSTANCE_ID_ELEMENT_NAME => 'measObjInstId'
```

- ASN1_HEADER_INFO_FOR_PIF_FILENAME: Values of PIF header entries can be included in the output PIF filename by including their element names in this array. It is critical that enough header counters are included here, so as to ensure a unique PIF file.

```
ASN1_HEADER_INFO_FOR_PIF_FILENAME=>['Type','StartDate','StartTime'
]
```

- ASN1_SUSPECT_FLAG_ELEMENT_NAME: This scalar entry contains the name of the element tag whose value contains the indication of the quality of the scanned data. This flag defaults to FALSE in the case of reliable data, If it's set to TRUE, then the record is unreliable, and the current PIF record is omitted from the PIF file. It is an optional item in the ASN1.

```
ASN1_SUSPECT_FLAG_ELEMENT_NAME = "suspectFlag"
```

- ASN1_ALTERNATIVE_COUNTER_NULL_VALUE: This scalar entry allows for the substitution of a defined value wherever NULL values are.

```
ASN1_ALTERNATIVE_COUNTER_NULL_VALUE  => '-9999'
```

- ASN1_OUTPUT_BLOCK_ELEMENT_NAME: This scalar entry contains the element tag whose value will be used for the PIF block name in the PIF files. It is also the first item in the PIF filename.

```
ASN1_OUTPUT_BLOCK_ELEMENT_NAME => 'measObjInstId'
```

- ASN1_3GPP_FILENAME_VALIDATION: This entry consists of tokens of the 3GPP measurement results filename and the corresponding Perl Regular Expressions that matches the filename token. Further validation such as range checking is done for configured items. The follow entries are mandatory in the filename and must appear in the ASN1_3GGP rule configuration

```
ASN1_3GPP_FILENAME_VALIDATION => {
                Type      => '^(A|B|C|D).+$',
                StartDate => '^\w{1}(\d{8}).+$',
                StartTime => '^\w{1\d{8}\.(\d{4})[+-](\d{4}).+$',
}
```

  The following entries are optional in the filename:
  EndDate
  Endtime
  UniqueID

RC

- ASN1_OBJECT_INSTANCE_ID_DESCRIPTION: This hash entry facilitates the identification of Measurement Object Instance Id classes by providing a regular expression against which the supplied Measurement Object Instance Id can be compared.

```
ASN1_OBJECT_INSTANCE_ID_DESCRIPTION => {
    "MsPlatform_AmrPool"  =>
      `\'ManagedElement=(\d+)\,MsPlatform=(\d+)\,AmrPool=(\d+)\'',
    "MsPlatform_CcPool"   =>
       '\'ManagedElement=(\d+)\,MsPlatform=(\d+)\,CcPool=(\d+)\'',
…
}
```

- ASN1_ALT_OBJECT_INSTANCE_ID_NAMES: This hash entry facilitates the provision of alternative measurement Object Instance Id names for those classes that do not provide the full/correct names as required by the PIF files. Names provided by the measurement object instance ID will be replaced by those provided in the ASN1_ALT_OBJECT_INSTANCE_ID_NAMES entry.

```
ASN1_ALT_OBJECT_INSTANCE_ID_NAMES => {
     "MsPlatform_AmrPool"  =>
          [qw(ManagedElement MsPlatform AmrPool)],
     "MsPlatform_CcPool"   =>
          [qw(ManagedElement MsPlatform CcPool)],
```

- ASN1_OBJECT_INSTANCE_ID_MAPPING: This hash contains a RE, OBJECT_DESCRIPTION which is used to attempt to match the object instance, if it is not matched in ASN1_OBJECT_INSTANCE_ID_DESCRIPTION. Typically though, either this hash or ASN1_OBJECT_INSTANCE_ID_DESCRIPTION is used for extraction. Once the object description has been matched any elements in the OBJECT_MAPPING hash are attempted to be mapped. If they are not found, a warning is logged and the null counter value is added. A sample is shown below:

```
'ASN1_OBJECT_INSTANCE_ID_MAPPING' => {
    'OBJECT_DESCRIPTION' => '\'(\w+)\..*\'',
    'OBJECT_MAPPING' => {
        'OBJ_ID' => '\'\w+\.(.*)\'',
    }
},
```

- ASN1_COPY_HEADER_TO_DATA_RECORDS: This mandatory entry contains an array entry of header counters that should be copied to each record for the corresponding PIF record block.

```
ASN1_COPY_HEADER_TO_DATA_RECORDS =>
          [qw(senderName_ManagedElement)],
```

2.2 ASN1_3GPP_INTERFACE

The essential task of the ASN1_3GPP_INTERFACE engine is to parse the vendor specific 3GPP ASN.1-compliant data into output PIF files.

There will typically be a one-to-many relationship between one ASN.1 file, which will probably map to several PIF files. One PIF file will be produced for every measInfo ASN.1 tag element in the data, as this ASN.1 block represents the counter names and values for a particular network abstraction for a particular measurement period. Note that the PIF file output will also include the header data common to the ASN.1 file.

A key goal of a successful configuration is to ensure that all the ASN.1 data is output to PIF files, and that no PIF duplication occurs. This duplication may occur if the configurations such as the PIF filename tokens do not give a unique filename for that particular measurement.

2.2.1 The asn1dump Extract Tool

The ASN1_3GPP_INTERFACE rule uses a pre-compiled asn1dump extract tool to facilitate ASN.1 decoding and parsing. The asn1dump binary reads ASN.1 encoded data, decodes the data, and dumps it to an ASCII file in a format that can be read and parsed by the ASN1_3GPP_INTERFACE rule.

An asn1dump pre-compiled binary is provided for each of the supported platforms.

2.2.2 Rule Configuration

The following details the vendor specific configuration rule entries for this Gateway.

- ASN1_EXTRACT_TOOL: The dumpasn1 tool is used to convert ASN.1 binary data into ascii. The ASN1_EXTRACT_TOOL entry is used to point to where the extraction tool is located.

```
ASN1_EXTRACT_TOOL   => '../parsersrc/PA-RISC2.0/dumpasn1'
```

Three different executables exist in the kit, for each of the platforms supported. By default the following extract tool is used:

```
ASN1_EXTRACT_TOOL   => '$VENDOR_GATEWAY/bin/asn1dump'
```

- ASN1_EXTRACT_TOOL_PARAMETERS: The ASN1_EXTRACT_TOOL_PARAMETERS entry is used to pass parameters to the ASN1_EXTRACT_TOOL.

```
ASN1_EXTRACT_TOOL_PARAMETERS   => '-p -c../parsersrc/dumpasn1.cfg'
```

By default, no parameters are passed.

- ASN1_ENABLE_EXTRACT_TO_TEMP_FILE: This optional flag indicates whether or not the ASN1_EXTRACT_TOOL should output the ASCII dump to a temporary file. Temporary files are concatenated with a '.tmp' extension.

```
ASN1_ENABLE_EXTRACT_TO_TEMP_FILE => "True"
```

- ASN1_DATA_FORMAT: The ASN1_DATA_FORMAT specifies the format of the ASN.1 data to be parsed. The configuration is implemented as a series of nested arrays.

```
ASN1_DATA_FORMAT => [
    [
        'fileFormatVersion','senderName','senderType',
        'vendorName','collectionBeginTime'
    ], [
        ['nEUserName','nEDistinguishedName'],
        [
            'measTimeStamp','granularityPeriod','measType',
            [
                'measObjInstId',
                ['iValue','rValue','noValue'],
                'suspectFlag'
            ],
        ],
    ],
    'measFileFooter',
]
```

- ASN1_PIF_HEADER_ELEMENT_NAMES: This scalar or array entry facilitates the inclusion of data in the output PIF header block.

```
ASN1_PIF_HEADER_ELEMENT_NAMES => [
    'fileFormatVersion',
    'senderName',
```

```
    'senderType',
    'vendorName',
    'collectionBeginTime',
    'nEUserName',
    'nEDistinguishedName',
    'measTimeStamp',
    'granularityPeriod'
]
```

This data is common to all PIF output files.

- ASN1_HEADER_INFO_FOR_PIF_FILENAME: This scalar or array entry facilitates the inclusion of data from the ASN1 header data in the output PIF filename.

```
ASN1_HEADER_INFO_FOR_PIF_FILENAME => [
    'NODEID',
    'START_DATE',
    'START_TIME',
    'nEDistinguishedName',
    'TYPE'
]
```

- ASN1_UNIQUE_PIF_BLOCK_ELEMENT_NAMES: This scalar entry identifies which element in the data file marks the beginning of data to be included in a new PIF file.

```
ASN1_UNIQUE_PIF_BLOCK_ELEMENT_NAMES => 'measTimeStamp'
```

- ASN1_COUNTER_NAMES: This scalar entry identifies which elements in the ASN1 data contain the counter names to be included in the PIF records.

```
ASN1_COUNTER_NAMES => 'measType'
```

- ASN1_COUNTER_VALUES: This scalar entry identifies which elements in the ASN1 data contain the counter values to be included in the PIF records.

```
ASN1_COUNTER_VALUES => '\w+Value'
```

- ASN1_3GPP_FILENAME_VALIDATION: The ASN1_3GPP_FILENAME_VALIDATION entry consists of tokens of the 3GPP measurement results filename and the corresponding Perl Regular Expressions that matches the filename token. Further validation such as range checking is carried out for configured items.

The follow entries are mandatory in the filename and must appear in the ASN1_3GPP_INTERFACE rule configuration.

```
ASN1_3GPP_FILENAME_VALIDATION => {
    Type      => '^(A|B|C|D).+$',
    StartDate => '^\w{1}(\d{8}).+$',
    StartTime => '^\w{1}\d{8}\.(\d{4}).+$',
}
```

The following entries are optional in the filename :

- Utc
- EndDate
- Endtime
- UniqueID
- RC

Validation Regular Expressions entries can be included in ASN1_3GPP_FILENAME_VALIDATION for optional items in the filename.

- ASN1_OUTPUT_BLOCK_ELEMENT_NAME: This scalar entry contains the element whose value will be used for the PIF block name in the PIF files. It is also the first item in the PIF filename.

```
ASN1_OUTPUT_BLOCK_ELEMENT_NAME => 'measObjInstId'
```

- ASN1_OBJECT_INSTANCE_ID_ELEMENT_NAME: This scalar entry contains the name of the element whose value contains the Measurement Object Instance Id. This scalar entry allows for the ID of the current measured object to be included in each record.

```
ASN1_OBJECT_INSTANCE_ID_ELEMENT_NAME => 'measObjInstId'
```

- ASN1_SUSPECT_FLAG_ELEMENT_NAME: The ASN1_SUSPECT_FLAG_ELEMENT_NAME scalar entry contains the name of the element whose value contains the indication of the quality of the scanned data. This flag defaults to FALSE in the case of reliable data. If it's set to TRUE, then the record is unreliable, and the current PIF record is omitted from the PIF file.

```
ASN1_SUSPECT_FLAG_ELEMENT_NAME => 'suspectFlag'
```

- ASN1_ALTERNATIVE_COUNTER_NULL_VALUE: This scalar entry allows for the substitution of a defined value where ever NULL values are encountered.

```
ASN1_ALTERNATIVE_COUNTER_NULL_VALUE  => '-9999'
```

- ASN1_HEX_TO_DECIMAL_ELEMENTS: This array entry allows the names of those elements whose values are presented as HEX values in the ASN.1 data file to be converted to decimal.

```
ASN1_HEX_TO_DECIMAL_ELEMENTS => ['granularityPeriod', 'iValue']
```

- ASN1_HEX_TO_CHAR_ELEMENTS: This array entry allows the names of those elements whose values are presented as HEX values in the ASN.1 data file to be converted to characters.

```
ASN1_HEX_TO_CHAR_ELEMENTS => ['senderType']
```

- ASN1_CHARACTER_SUBSTITUTION: This hash entry facilitates pre-parser character substitution mappings. The hash keys contains the current character ASCII values and the corresponding values contains their replacements.

  The following example is the German special character replacements:

```
CHARACTER_SUBSTITUTION => {
    "\337" => "ss",
    "\344" => "ae",
    "\366" => "oe",
    "\374" => "ue",
    "\304" => "Ae",
    "\326" => "Oe",
    "\334" => "Ue",
}
```

- ASN1_OBJECT_INSTANCE_ID_DESCRIPTION: This hash entry facilitates the identification of Measurement Object Instance Id classes by providing a regular expression against which the supplied Measurement Object Instance Id can be compared.

```
ASN1_OBJECT_INSTANCE_ID_DESCRIPTION => {
    "HNDOVER"    => '^HNDOVER\.(\w*.*)\'',
}
```

- ASN1_ALT_OBJECT_INSTANCE_ID_NAMES: This hash entry facilitates the provision of alternative measurement Object Instance Id names for those classes that do not provide the full/correct names as required by the PIF files. Names provided by the measurement object instance id will be replaced by those provided in the ASN1_ALT_OBJECT_INSTANCE_ID_NAMES entry.

```
ASN1_ALT_OBJECT_INSTANCE_ID_NAMES => {
```

```
    "M3ASPSM"    => ['X_ID1'],
    "M3ASPTM"    => ['X_ID1'],
    "M3ASSOC"    => ['X_ID1'],
    "M3DATA"     => ['X_ID1'],
    "M3MGMT"     => ['X_ID1'],
    "M3SSNM"     => ['X_ID1'],
}
```

Each array entry corresponds to an extraction () in the corresponding regular expression entry in the ASN1_OBJECT_INSTANCE_ID_DESCRIPTION for each class Id.

- ASN1_OBJECT_INSTANCE_ID_MAPPING: This hash contains a RE, OBJECT_DESCRIPTION which is used to attempt to match the object instance, if it is not matched in ASN1_OBJECT_INSTANCE_ID_DESCRIPTION. Typically though, either this hash or ASN1_OBJECT_INSTANCE_ID_DESCRIPTION is used for extraction. Once the object description has been matched any elements in the OBJECT_MAPPING hash are attempted to be mapped. If they are not found, a warning is logged and the null counter value is added. A sample is shown below:

```
ASN1_OBJECT_INSTANCE_ID_MAPPING => {
    'OBJECT_DESCRIPTION' => '\'(\w+)\..*\'',
    'OBJECT_MAPPING' => {
        'OBJ_ID' => '\'\w+\.(.*)\'',
    }
}
```

- ASN1_OBJECT_INSTANCE_ID_OFFSET_MAPPING: This hash contains two items. A string OBJECT_DESCRIPTION which defines the counter name that will contain the mapping value. OBJECT_MAPPING is a hash that consists of <BLOCKNAME>:<OFFSET> mappings. Elements are matched against the OBJECT_MAPPING hash to determine whether the counter defined by OBJECT_DESCRITPION is required. A running counter keeps track of the number of <BLOCKNAME> records occurring in the current raw file. The counter starts at zero (0) and is reset when processing a new raw file. If a mapping exists, the running counter for the particular <BLOCKNAME> + <OFFSET> is assigned as the OBJECT_DESCRIPTION counter value. A sample of the configuration is shown below:

```
ASN1_OBJECT_INSTANCE_ID_OFFSET_MAPPING => {
    'OBJECT_DESCRIPTION' => 'OBJ_ID',
    'OBJECT_MAPPING' => {
        'SS7HSLPG1' => 8,
        'SS7HSLPG2' => 8,
    }
```

```
}
```

Given the example above, if there are multiple SS7HSLPG1 records, the first SS7HSLPG1 record in the raw file will be assigned an OBJ_ID with the running counter value for SS7HSLPG1 + its OFFSET (8). OBJ_ID=8 (0 + OFFSET). The next SS7HSLPG1 record will have OBJ_ID=9 (1 + OFFSET), so on so forth until all records have been processed and assigned a counter value.

3. Post Parser Rules and Configuration

The following section describes the post parser rules, which are provided together with this parser package.

3.1 TRANSPLANT_HEAD

This post parser rule allows counters to be copied from the header record to each record in the data block of the PIF file. The counters that are going to be copied shall be listed out in the specific array.

The non-standard configuration entries with this rule are listed below.

3.1.1 Rule Configuration

- HEADER_COUNTERS_FOR_TRANSPLANT: A mandatory field that consists of an array of header counters. These counters shall be copied to the PIF body.

```
HEADER_COUNTERS_FOR_TRANSPLANT => [qw(senderName_MeContext)]
```

A full sample configuration is included below:

#Add configuration info in vmgw block

```
{
    RULE_TYPE                       => 'TRANSPLANT_HEAD',
    RULE_DESC                       => 'Transplant Header information
                                        into the Vmgw Data records',
    INPUT_FILE_DESCRIPTION => ["^MgwApplication_Vmgw-#-.*-#-I.pif"],
    HEADER_COUNTERS_FOR_TRANSPLANT => [qw(senderName_MeContext)],
    PRODUCE_PIF                     => 'True',
    PRODUCE_LIF                     => 0,
},
```

4. Appendix A - 3GPP ASN1 Description Details

From [3GPP TS 32.401], a 3GPP compliant ASN1 BER encoded file shall conform to the ASN1 file format definition shown below:

```
PM-File-Description
DEFINITIONS AUTOMATIC TAGS::= BEGIN

MeasDataCollection::= SEQUENCE
      {
      measFileHeader          MeasFileHeader,
      measData                SEQUENCE OF MeasData,
      measFileFooter          MeasFileFooter
      }

MeasFileHeader::= SEQUENCE
      {
      fileFormatVersion             PrintableString (SIZE (0..15)),
      senderName                    PrintableString (SIZE (0..400)),
      senderType                    SenderType,
      vendorName                    PrintableString (SIZE (0..32)),
      collectionBeginTime           TimeStamp,
      ...
      }
-- The sole purpose of the ellipsis notation used in the file header is
to facilitate inter-release compatibility, vendor specific additions are
not allowed in implementations claiming conformance to the TS. However,
it is acknowledged that this feature does enable the use of non-standard
extensions to the file header without loosing compatibility to the file
format specified in the present document.

SenderType::= PrintableString (SIZE (0..8))

TimeStamp::= GeneralizedTime

MeasData::= SEQUENCE
      {
      nEId              NEId,
      measInfo          SEQUENCE OF MeasInfo
      }

NEId::= SEQUENCE
      {
      nEUserName              PrintableString (SIZE (0..64)),
      nEDistinguishedName     PrintableString (SIZE (0..400)),
      nESoftwareVersion       PrintableString (SIZE (0..64)) OPTIONAL
      }

MeasInfo::= SEQUENCE
      {
      measTimeStamp           TimeStamp,
      jobId                   [1]   INTEGER OPTIONAL,
      granularityPeriod       [2]   INTEGER,
      reportingPeriod         [3]   INTEGER OPTIONAL,
      measTypes               [4]   SEQUENCE OF MeasType,
```

```
      measValues              SEQUENCE OF MeasValue
      }

MeasType::= PrintableString (SIZE (1..32))

MeasValue::= SEQUENCE
      {
      measObjInstId           MeasObjInstId,
      measResults             SEQUENCE OF MeasResult,
      suspectFlag             BOOLEAN DEFAULT FALSE
      }

MeasObjInstId::= PrintableString (SIZE (0..64))

MeasResult::= CHOICE
      {
      iValue      INTEGER,
      rValue      REAL,
      noValue     NULL,
      ...
      }

-- Normal values are INTEGERs and REALs. The NULL value is reserved to
indicate that the measurement item is not applicable or could not be
retrieved for the object instance. The sole purpose of the ellipsis
notation used in the MeasResult choice is to facilitate inter-release
compatibility in case the choice needs to be extended in future
releases.

MeasFileFooter::= TimeStamp

END
```