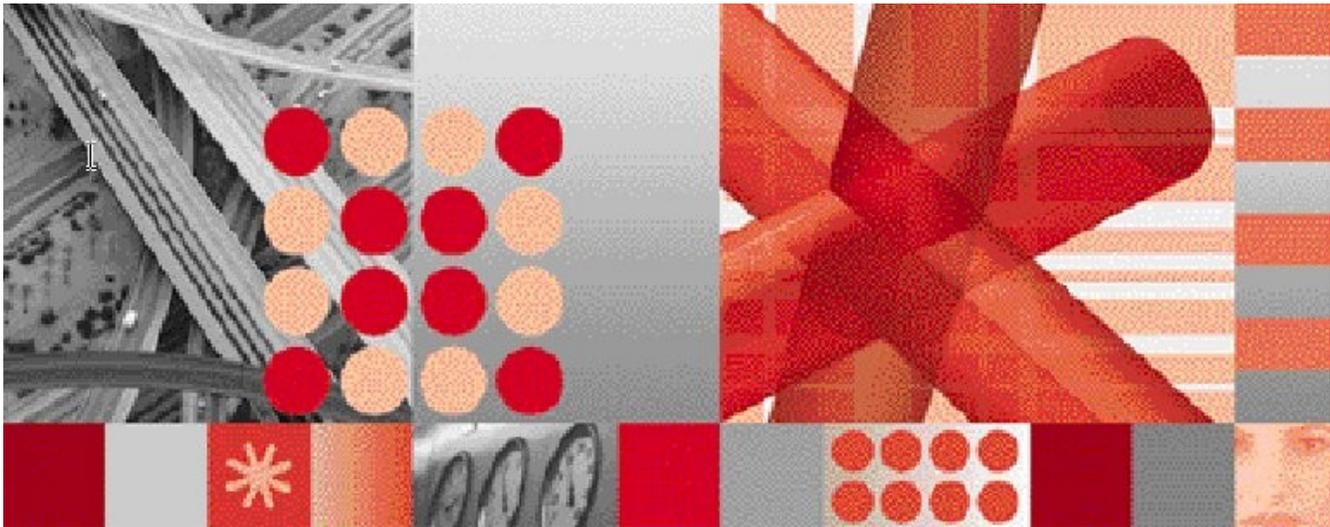


Version 3.5.0



Nokia XML Gateway User Guide

Note: Before using this information and the product it supports, read the information in Notices and Trademarks on page 14.

This edition applies to Version 3.5 of IBM® Tivoli® Netcool® Gateway Framework and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation, 2011. All rights reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Table of Contents

1 About This Documentation	1
1.1 Audience.....	1
1.2 Required Skills and Knowledge.....	1
2 Overview	2
2.1 The Gateway Framework.....	2
2.2 Nokia XML Overview.....	2
2.2.1 Nokia XML.....	2
2.2.2 Network Details.....	2
2.2.3 Data Types.....	2
2.2.4 Data/File Formats.....	3
2.2.5 Performance Data Layout.....	3
2.2.6 File naming specification.....	3
2.2.7 Architectural extensions.....	4
2.2.8 DTD Validation.....	4
3 Engine Rules and Configuration	5
3.1 Nokia_XML.....	5
3.1.1 Processing Sequence.....	5
3.1.2 Rule Configuration.....	5
3.2 Nokia_XML_CONFIGURATION.....	7
3.2.1 Rule Configuration.....	7
4 Post Parser Rules and Configuration	9
4.1 COUNTER_JOIN.....	9
4.1.1 Rule Configuration.....	9
4.2 COPY_HEADER_N_BODY_COUNTERS.....	10
4.2.1 Rule Configuration.....	10
5 Installation Specific Information	12
Appendix A : Sample Nokia XML	13
Notices and Trademarks	15

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.
Nokia OSS 4.0 Measurement Data Export Interface	<p>This document explains how the Measurement Data Export from Nokia NetAct to an external system works in principle.</p> <p>The Measurement Data Export interface is a solution for publishing measurement data in Open Measurement Standard (OMeS) format.</p>

Glossary

OMeS	Open Measurement Standard
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

1 About This Documentation

1.1 Audience

The target audience of this document is IBM Performance Manager for Wireless customers. They should be familiar with telecommunication and IT principles and should also have a good understanding of Solaris.

IMPORTANT: Before attempting an installation of Performance Manager for Wireless you are strongly advised to read the release notes and any readme files distributed with your Performance Manager for Wireless software. Readme files and release notes may contain information specific to your installation not contained in this guide. Failure to consult readme files and release notes may result in a corrupt, incomplete or failed installation.

Note: Performance Manager for Wireless Administrators should not, without prior consultation and agreement from IBM, make any changes to the Index Organized tables or database schema. Changes to the Index Organized tables or database schema may result in corruption of data and failure of the Performance Manager for Wireless System. This applies to all releases of Performance Manager for Wireless using all versions of interfaces.

1.2 Required Skills and Knowledge

This guide assumes you are familiar with the following:

- General IT Principles
- Sun Solaris Operating System
- Oracle Database
- Windows operating systems
- Graphical User Interfaces
- Network Operator's OSS and BSS systems architecture

This guide also assumes that you are familiar with your company's network and with procedures for configuring, monitoring, and solving problems on your network.

2 Overview

2.1 The Gateway Framework

The Nokia XML uses the Gateway Framework as a container for the execution of its engine and post parser stages. The Gateway Framework and Nokia XML 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 Nokia XML 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 Nokia XML Gateway plugs into the Gateway Framework and extends this functionality to provide the final Gateway that parses the Nokia XML data. More information on the standard Gateway configuration is contained in the [Gateway Framework User Guide].

The Nokia XML Gateway supports the parsing of performance measurements data from Nokia 3G networks. The configuration of this interface is designed to be highly configurable and extensible into the future.

Only Nokia XML Gateway specific configuration details will be described in this document.

2.2 Nokia XML Overview

2.2.1 Nokia XML

Nokia uses a standard known as the Open Measurement Standard (OMeS) for transferring measurement data between Nokia NetAct™ and external systems. The OMeS file format is a proprietary standard defined by Nokia using Extensible Markup Language (XML). The OMeS XML will be referred to as Nokia XML throughout this document interchangeably.

Nokia NetAct™ is a complete solution for managing mobile networks and end-user services, ranging from control-room software for 24/7/365 monitoring of the network to software for graphical optimization. The release of this vendor gateway is to include the parsing of data output from Nokia NetAct™ OSS4.0 for OMeS versions 1.0 and 2.0 XML files.

2.2.2 Network Details

The XML performance measurement data derives from GPRS or UMTS network elements, typically via the OA&M management collection point for the network. The periodic measurement files, which can be sizeable, are delivered by the Network OA&M system for collection by northbound OSS applications. This delivery mechanism simplifies the Gateway interface with the network to that of a synchronous file transfer at predefined intervals.

2.2.3 Data Types

Measurement data is collected from the NEs to NetAct. These measurements include several areas of performance evaluation including user and signalling traffic, verification of the network configuration, resource access measurements, Quality of Service (e.g. delays during call set-up, packet throughput, etc) and resource availability.

The Nokia XML Gateway processes two basic data types, the performance data and configuration data.

2.2.4 Data/File Formats

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

2.2.5 Performance Data Layout

The generated performance measurement data shall be in ASCII format and in a OMeS XML structure.

The OMeS data model consists of three basic elements that identify and describe the measurements and the measurement results:

1. *Setup* identifies and describes the time stamp of the measurement and the measurement interval. The OMeS 1.0 setup also describes the network element (NE) that usually controls the measurement.
2. *Target* identifies the measured object (MO). Measurement Data Export interface supports distinguished names (DN) only.
3. *Performance Indicator* identifies the performance indicator (PI) that is measured.

OMeS XML files contain information about the measured object, the starting time and the measurement period, the duration of the period and the actual counter data.

A sample Nokia XML file is shown in Appendix A.

2.2.6 File naming specification

The naming convention for the export files are as shown below:

etlexpmx_<firstMeasuredObjectClass>_<timeStamp>_<uniqueID>.xml

For example: ***etlexpmx_MMSC_20050208174900_100.xml***

- <firstMeasuredObjectClass> refers to the measured object class of the first record found from the OMeS file received from the Network Element.
- <timeStamp> is the date and time when the granularity period begins or the first granularity period of the measurement result comes to availability. It takes the format of YYYYMMDDHHmmss, where
 - YYYY - 4 digits year.
 - MM - 2 digits month (01-12).
 - DD - 2 digits day (01-31).
 - HH - 2 digits hour (00-23)

- mm - 2 digits minute (00-59)
- ss - 2 digits second (00-59)
- <uniqueID> is the name of the NE, EM or domain. This field is optional.

Example of Nokia XML performance data files are as follows:

```
etlexpmx_AXC_20050915084650_159.xml  
etlexpmx_RNC_20050915165840_194.xml  
etlexpmx_WCEL_20050914155329_111.xml
```

2.2.7 Architectural extensions

The Nokia XML Gateway parses OMeS XML data file to the Gateway PIF format. The interface uses the XML::Parser parser library to facilitate XML parsing. XML DTD and Schema validation is not currently supported. The core 3rd party tools are the XML::Parser Perl modules which interfaces to the Expat C based XML parser.

The XML::Parser XML parser complies with SAX 1.0 and 2.0 specifications. The Nokia_XML gateway uses SAX callback routines to handle XML elements and attributes, etc.

2.2.8 DTD Validation

No DTD validation is supported in the Gateway. Deployments requiring such validation should seek to have this process completed by the equipment vendor element management systems supplying the XML.

3 Engine Rules and Configuration

3.1 Nokia_XML

The essential task of the Nokia_XML engine is to parse the XML data into output PIF files. This gateway uses its configuration to extract the data required for the performance tool.

3.1.1 Processing Sequence

The Nokia_XML processes the performance data by parsing the XML file for header counters and data counters.

Header counters are parsed based on the object elements and attributes defined within the **XML_PIF_HEADER_ELEMENT_NAMES** and **XML_UNIQUE_PIF_BLOCK_HEADER_ATTRIBUTE_NAMES** entries.

Unique PIF files are created for each PIF mapping defined within **XML_HEADER_INFO_FOR_PIF_FILENAME**.

All the data counters are parsed for each data block defined within the entry **XML_UNIQUE_PIF_BLOCK_ELEMENT_NAMES**. Counter names and counter values are mapped to a single row in the PIF file.

3.1.2 Rule Configuration

The following details the vendor specific rule entries for the Nokia XML engine rule.

- **XML_HEADER_DATE_TIME_DESCRIPTION**: This entry consists of tokens of the measurements results attribute name and the corresponding Perl Regular Expressions that matches the attribute name token. Each token will be validated and recorded as the header counter.

```
XML_HEADER_DATE_TIME_DESCRIPTION => {  
    StartDate => '^(\d{4}-\d{2}-\d{2}).*$',  
    StartTime => '^.+T(\d{2}:\d{2}).*$',  
}
```

- **XML_PIF_HEADER_ELEMENT_NAMES**: This scalar or array entry facilitates the inclusion of XML element data in the output PIF header block. This data is common to all PIF output files.

For OMeS 2.0, where DN is an element instead of an attribute:

```
XML_PIF_HEADER_ELEMENT_NAMES => [ 'DN' ],
```

- **XML_UNIQUE_PIF_BLOCK_ELEMENT_NAMES**: These elements contain the data blocks to be parsed into a unique PIF file.

For OMeS 1.0:

```
XML_UNIQUE_PIF_BLOCK_ELEMENT_NAMES => [ 'BG_Nokia_BG',  
    'GNS_Nokia_GNS', 'FW_Nokia_FW', ],
```

For OMeS 2.0:

```
XML_UNIQUE_PIF_BLOCK_ELEMENT_NAMES => [ 'PMMOResult' ],
```

Note: OMeS 2.0 uses PMMOResult as the standard element name for data blocks within the XML files. Therefore PMMOResult is used exclusively for OMeS2.0 XML data file as the

unique PIF block element name.

- XML_UNIQUE_PIF_BLOCK_HEADER_ATTRIBUTE_NAMES: This scalar or array entry facilitates the inclusion of attribute within each XML element data in the output PIF header block. This data is common to all PIF output files.

For OMeS 1.0:

```
XML_UNIQUE_PIF_BLOCK_HEADER_ATTRIBUTE_NAMES =>
  ['DN', 'measurementType', 'startTime', 'interval'],
```

For OMeS 2.0:

```
XML_UNIQUE_PIF_BLOCK_HEADER_ATTRIBUTE_NAMES =>
  [ 'measurementType', 'startTime', 'interval'],
```

- XML_OUTPUT_BLOCK_NAME: This scalar entry allows for setting the default initial output block name.

```
XML_OUTPUT_BLOCK_NAME => 'Nokia',
```

- XML_OUTPUT_BLOCK_DESCRIPTION: This entry configures the element name to be included to the output block name.

- XML_OUTPUT_BLOCK_DESCRIPTION => '^(*)_Nokia_*',

Note: The expression will return the argument matched with the object element name obtained from XML_UNIQUE_PIF_BLOCK_ELEMENT_NAMES.

- XML_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.

```
XML_HEADER_INFO_FOR_PIF_FILENAME=>['Prefix',
  'StartDate',
  'StartTime',
  ],
```

There will typically be a one-to-many relationship between one XML file, which will probably map to several PIF files. The PIF filenames will have a prefix of the PIF data block name in addition to the unique combination of elements defined within the entry:

```
XML_HEADER_INFO_FOR_PIF_FILENAME
```

These elements represent the header data for a particular network abstraction for a particular measurement period.

A key goal of a successful configuration is to ensure that all the XML data is outputted in 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.

- XML_ENABLE_PIF_COUNTER_IN_FILENAME: This scalar entry enables the inclusion of a PIF counter 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.

```
XML_ENABLE_PIF_COUNTER_IN_FILENAME => 'True',
```

or

```
XML_ENABLE_PIF_COUNTER_IN_FILENAME => 0,
```

- HEADER_DATA_RECORD_PROCESSING: This is an optional entry whose value points to a configured
-

Perl subroutine. The purpose of this entry is to add custom processing for a deployment, such as add a new counter. But it could be any processing on the header and data record. This entry deals with counter name/value pairs in the current header record and is invoked for each header record encountered.

The Nokia_XML default rule configuration for OMeS 2.0 uses the

HEADER_DATA_RECORD_PROCESSING to perform the following tasks:

- Round down StartTime to the nearest interval, and
- Extract element ID from DN.

```
HEADER_DATA_RECORD_PROCESSING => sub {
    my ($parent_ref, $rec_ref, $op_hash_ref) = @_;
    # Rounding down StartTime to the nearest interval
    my ($hour, $minute) = split(":", $op_hash_ref->{StartTime});
    minute = $minute - ($minute % $op_hash_ref->{interval});
    $op_hash_ref->{StartTime} = sprintf("%02d:%02d", $hour, $minute);
    # Extract element ID from DN
    if ( $header_ref->{'measurementType'} =~ /.*/ ) {
        for ($data_ref->{DN}) {
            !$_ && last; # skip processing if not exists
            /(GPBB)\-(\d+)/ && do {$data_ref->{"$1_ID"}="$1-$2";};
            /(GPST)\-(\d+)/ && do {$data_ref->{"$1_ID"}="$1-$2";};
            /(GGSN)\-(\d+)/ && do {$data_ref->{"$1_ID"}="$1-$2";};
        }
    }
    return 0;
}
```

3.2 Nokia_XML_CONFIGURATION

The processing steps of Nokia_XML_CONFIGURATION are similar to Nokia_XML. The only difference is that it is dealing with bulk configuration data that is coming from the network.

This gateway supports the parsing of network configuration data from Nokia UMTS networks. The configuration of this gateway is designed to be highly configurable and extensible into the future.

The gateway is designed to convert from Nokia UMTS XML configuration data to the Gateway PIF format.

3.2.1 Rule Configuration

- IGNORE_ELEMENTS: This array entry contains the list of elements that should be ignored by the rule. All sub-elements within the element will also be ignored.

```
IGNORE_ELEMENTS => [],
```

- COMMON_HEADER_DATA_OBJECTS: This array entry lists the atomic element names, which should be included in the PIF header as counters. These entries are outside the OBJECT_MAPPING XML.
- OBJECT_MAPPING: This mandatory hash entry contains the basic object mapping of the XML to PIF. Each OBJECT_MAPPING hash key contains the name of the XML element that commences a new PIF output for the contents of that element. The id attribute for this element is placed in the header of the PIF. Each OBJECT_MAPPING key entry in turn references a hash containing configuration entries, which determine certain mappings for the element in question. These configuration entries are:

- **PIF_FILENAME_KEYS**: This array entry lists the counter names in either the header or record data that should be used in formulating the PIF filename.
- **CHILD_ELEMENTS**: This hash entry contains the PIF record mapping details. Each key contains an element name that maps to the point in the XML when a new PIF record commences. Each key references another hash entry that contains configuration information relevant to the PIF record. These configuration entries are:
 - **NON_UNIQUE_CLASS_BLOCKS**: This array entry contains the names of elements within the child element, which appear more than once. Because of the potential for overwriting on these element counters, these counter entries in the PIF filename are pre-pended with a tag. This tag is made up of the **NON_UNIQUE_CLASS_BLOCKS** element name and its "id" attribute. **NON_UNIQUE_CLASS_BLOCKS** entries must have an "id" attribute in order to ensure uniqueness.
 - **ENABLE_UNIQUE_PIF**: Setting the **ENABLE_UNIQUE_PIF** to "TRUE" facilitate the placing of each PIF record in a unique PIF file.
 - **OUTPUT_BLOCK_NAME**: This scalar entry allows for setting the name of the output block name for a particular **CHILD_ELEMENTS** entry.
 - **DUPLICATE_COUNTERS**: An atomic element name, appearing as an actual counter in the PIF file should be included in this array list, if the name of the counter appears more than once in the XML block for the PIF record. This facilitates the placing of a unique incrementing numeric value on these values (vsDataType_1, vsDataType_2 etc.).

A sample **OBJECT_MAPPING** entry is as follows:

```
OBJECT_MAPPING => {  
  'WBTS_1.5' => {  
    PIF_FILENAME_KEYS => [qw( WCEL_1.5_id )],  
    CHILD_ELEMENTS => {  
      'WCEL_1.5' => {  
        NON_UNIQUE_CLASS_BLOCKS => [qw(ADJS_1.5)],  
        ENABLE_UNIQUE_PIF => 0,  
      },  
    },  
  },  
},
```

4 Post Parser Rules and Configuration

4.1 COUNTER_JOIN

This rule creates new header and data counters based on the joining of a number of counter values to create new composite keys. For example for 2 counters PLMN_NAME -> 2 and RNC_NAME -> 3 a new counter could be created called PLMN_RNC value 2/3. The non standard configuration options for this rule are:

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

4.1.1 Rule Configuration

- **HEADER_COUNTERS_TO_JOIN**: An array of anonymous hashes detailing the header counters to join together to create new counters.

Each entry contains 3 configuration entries:

- **COUNTERS_TO_JOIN**: This array lists the header counters to join together
- **NEW_COUNTER_NAME**: The name to assign to the new counter created.
- **COUNTER_SEPARATORS**: The characters to use to separate the counters in the newly created value. Typically they will be separated by a "-" but any string can be configured to allow "-" and "/" to be mixed etc.

Note: If there are N counters to join then N-1 separators need to be configured.

```
{  
  COUNTERS_TO_JOIN => [  
    qw(PLMN_NAME RNC_NAME)  
  ],  
  NEW_COUNTER_NAME => 'PLMN_RNC',  
  COUNTER_SEPARATORS => ['-'],  
},
```

- **DATA_COUNTERS_TO_JOIN**: This array of anonymous hashes is configured in an identical way to the **HEADER_COUNTERS_TO_JOIN** except it specifies the list of data counters to join.

See above for the configuration entries.

- **DEFAULT_NULL_VALUE**: The value to use for the new counter if the counters specified to create the new counter don't exist in the input.

A full configuration of the rule is below:

```
{  
  RULE_TYPE=>'COUNTER_JOIN',  
  RULE_DESC=>'Join counter values to create composite key',  
  INPUT_FILE_DESCRIPTION=>'.*RNC.*CE\.pif',  
  OUTPUT_FORMAT=>'LIF_Writer',  
  HEADER_COUNTERS_TO_JOIN=>[  
    {  
      COUNTERS_TO_JOIN => [  
        qw(PLMN_NAME RNC_NAME)  
      ],  
      NEW_COUNTER_NAME => 'PLMN_RNC',  
      COUNTER_SEPARATORS => ['-'],  
    }  
  ],  
  DATA_COUNTERS_TO_JOIN => [  
    {  
      COUNTERS_TO_JOIN => [  
        qw(RNC_NAME CELL_ID)  
      ],  
      NEW_COUNTER_NAME => 'RNC_CELL_ID',  
      COUNTER_SEPARATORS => ['-'],  
    }  
  ],  
  DEFAULT_NULL_VALUE => '0',  
}
```

```

        COUNTERS_TO_JOIN=>[qw(PLMN_NAMERNC_NAME)],
        NEW_COUNTER_NAME=>'PLMN_RNC',
        COUNTER_SEPARATORS=>['-'],
    },
],
DATA_COUNTERS_TO_JOIN=>[
    {
        COUNTERS_TO_JOIN=>[
            qw(WCEL_NAMEWBTS_NAMEWBTS_ID)
        ],
        NEW_COUNTER_NAME=>'WCEL_WBTS',
        COUNTER_SEPARATORS=>[qw(-/)],
    },
],
DEFAULT_NULL_VALUE=>'NULL'
}

```

4.2 COPY_HEADER_N_BODY_COUNTERS

This post parser rule allows counters to be copied from header to body, and vice versa. 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:

4.2.1 Rule Configuration

- **COPY_HEADER_COUNTERS_2_BODY**: A mandatory field that consists of an array of header counters. These counters shall be copied to the PIF body.

```
COPY_HEADER_COUNTERS_2_BODY => [ 'RNC_RNCName', 'RNC_RNCIdentifier', ],
```

- **COPY_BODY_COUNTERS_2_HEADER**: A mandatory field that consists of an array of PIF body record counters. These counters shall be copied to the PIF header.

```
COPY_BODY_COUNTERS_2_HEADER =>
[ 'NODEB_NODEBIPAddress', 'RNCCELL_RncNodeBIdentifier' ],
```

- **REDUNDANT_HEADER_COUNTERS**: An optional field that consists of an array of header counters. These counters shall be removed.

```
REDUNDANT_HEADER_COUNTERS => [ 'RNC_RNCName', 'RNC_RNCIdentifier', ],
```

- **REDUNDANT_BODY_COUNTERS**: An optional field that consists of an array of body record counters. These counters shall be removed.

```
REDUNDANT_BODY_COUNTERS => [ 'NODEB_NODEBIPAddress',
    'RNCCELL_RncNodeBIdentifier', ],
```

A full sample configuration is included below:

```

{
    'RULE_TYPE' => 'COPY_HEADER_N_BODY_COUNTERS',
    'RULE_DESC' => 'Copy Header information to the Data records',
    'INPUT_FILE_DESCRIPTION' => [ 'PLMN.*NE_Specific.*-#-CE\.pif',
    'PLMN.*GGSN_AP.*-#-CE\.pif',

```

```
'PLMN.*Interface.*-#-CE\.pif'],  
'COPY_HEADER_COUNTERS_2_BODY' => ['INTERFACE_ID'],  
'COPY_BODY_COUNTERS_2_HEADER' => [ ],  
'REDUNDANT_HEADER_COUNTERS'=> ['INTERFACE_ID'],  
'REDUNDANT_BODY_COUNTERS' => [ ],  
'PRODUCE_PIF' => 'True',  
},
```

Appendix A : Sample Nokia XML

```
<?xml version='1.0'?>
<OMeS xmlns='pm/cnf_rnc_nokia_2.xsd'>
  <PMSetup DN = 'PLMN-PLMN/RNC1'
    startTime='2001-02-17T 15:00:00.000+02:00:00' interval='60'>
    <RNC_Nokia_1.2 localMoid='DN:unit_type-2/unit_index-0'
      measurementType='Unit_Load_Measurement'
      MOName='RNC-1_DN:unit_type-2/unit_index-0'
      <M250C0> 3</M250C0>
      <M250C1> 14</M250C1>
      <M250C2> 20</M250C2>
      <M250C3> 15</M250C3>
    </RNC_Nokia_1.2>
  </PMSetup>
  <PMSetup DN = 'RNC-2'
    startTime='2001-02-17T 16:00:00.000+02:00:00' interval='60'
    realStartTime='2001-02-17T16:01:23.000+02:00:00'
    realStopTime='2001-02-17T17:01:12.000+02:00:00'>
    <RNC_Nokia_1.2 localMoid='DN:unit_type-2/unit_index-1'
      measurementType='Unit_Load_Measurement'
      MOName='RNC-1_DN:unit_type-2/unit_index-1'
      <M250C0> 10</M250C0>
      <M250C1> 12</M250C1>
      <M250C2> 8</M250C2>
      <M250C3> 17</M250C3>
    </RNC_Nokia_1.2>
  </PMSetup>
</OMeS>
```

Table 1 – Sample OMeS 1.0 XML performance data file contents.

```
<?xml version="1.0"?>
<OMeS>
  <PMSetup startTime="2005-03-08T00:00:00.000+01:00:00" interval="60">
    <PMMOResult>
      <MO>
        <DN>PLMN-PLMN/ADAP-NOKGPRSTRA/NGSG-123456/NGCP-100/NGMS-24407</DN>
      </MO>
      <PMTarget measurementType="madgtrIMSI">
        <total>4</total>
        <success>4</success>
        <ecgr1>0</ecgr1>
        <ecgr2>0</ecgr2>
        <ecgr3>0</ecgr3>
        <ecgr4>0</ecgr4>
        <ecgr5>0</ecgr5>
        <ecgr6>0</ecgr6>
        <ecgr7>0</ecgr7>
        <ecgr8>0</ecgr8>
        <ecgr9>0</ecgr9>
        <ecgr10>0</ecgr10>
        <ecgr11>0</ecgr11>
        <ecgr12>0</ecgr12>
        <ecgr13>0</ecgr13>
        <ecgr14>0</ecgr14>
        <ecgr15>0</ecgr15>
        <ecgr16>0</ecgr16>
      </PMTarget>
    </PMMOResult>
  </PMSetup>
</OMeS>
```

Table 2 – Sample OMeS 2.0 XML performance data file contents.

Notices and Trademarks

This appendix contains the following:

- Notices
- Trademarks

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM

product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758
U.S.A.*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo and [ibm.com](http://www.ibm.com) are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.



Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product or service names may be trademarks or service marks of others.



Printed in the USA.