Ericsson GSM Gateway User Guide

Date: 29 January 2008

1 Associated Documents

The following documentation accompanies this release:

1.1 Referenced Documents

Document Name	**Document Description**
[Gateways Install Note]	This document describes the steps required to install and run a Gateway.

1.2 Other Related Documents

Document Name	**Document Description**
[Gateway Framework User Guide]	Gateway Framework User Guide describing the management and configuration of the Gateway Framework.

1.3 Glossary

BSS	Base Station Subsystem
PIF	Parser Intermediate Format
LIF	Loader Input Format

2 Overview

2.1 The Gateway Framework

The Ericsson BSS Gateway, also refered to as the Vendor Gateway, uses the Gateway Framework as a container for the execution of its engine and post parser stages. The Gateway Framework and Vendor 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 Vendor 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 Vendor Gateway plugs into the Gateway Framework and extends this functionality to provide the final Gateway that parses the vendor data.
More information on the standard Gateway configuration is contained in the Gateway Framework User Guide.
Only vendor specific configuration details will be described in this document.

2.2 Ericsson GSM Gateway Overview

2.2.1 Network Details

The Ericsson Statistics and Traffic Measurement Subsystem (STS) provides OMC-R data records and radio network configuration parameters reports.

2.2.2 Data Version Support

Data versions supported by this vendor gateway are v9.1, v10, and v11 Ericson BSS raw and configuration data.

3 Engine Rules and Configuration

The following are 10 configurable Engine rules for the Ericsson GSM Gateway.

Ericsson Interface	Engine Rule Name
ASCII	ASCII_INTERFACE
C7PM	R2P_C7PM
MTCES	R2P_MTCES
MTRRCR	R2P_MTRRCR
MTRVC	R2P_MTRVC
SEQS	R2P_SEQS
STFIOP	R2P_STFIOP
TRAR	R2P_TRAR
TRART	R2P_TRART
TRDIP	R2P_TRDIP

3.1 Ascii Engine Rule

This Engine Rule allows you to process Ericsson ascii files.
This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory -	'ASCII_INTERFACE' - The string the Gateway uses to identify the type of this rule.
JUNK_MATCH	optional -	A list of regular expressions describing the lines of the input file that are to be discarded before parsing begins. Typical usage of this option is to exclude formatting information such as page headers from the report.
BAD_CHARS	optional -	A list of characters to remove from the input before parsing. The most common use of this is to remove line-feed, form-feed and null characters which can sometimes be present in a report. This list should always contain at least the null character (specified with '\000') as these can be hidden in an editor such as vi but may cause the parser to behave incorrectly.
BLANK_MATCH	mandatory -	A regular expression matching lines in the input that are empty of any report information. The normal setting should be '^[\t]*$' which will match lines containing only spaces and tabs, or those that are completely empty. Multi-object and 2-level layouts rely upon whitespacing to determine certain important features of a report.
SUB_SPACES	mandatory -	The character to replace spaces within counter names, normally this would be '_'. (required)
REPORT_END	mandatory -	Regular expression describing the match required for a report end. This expression should describe at least the start of the line as well as the actual text. 'END' alone might inadvertently match a counter name and cause truncation of a report. By using '^\s*END' only lines starting with whitespace

		followed by 'END' will match. (required)
REPORT_TYPES	mandatory -	A list containing the report type specific configuration rules, there are further details of its options in the next section.

Example :
```
      {
                 'RULE_TYPE' => 'ASCII_INTERFACE',
                 'RULE_DESC' => 'Convert Ericsson ASCII Report files
      to PIF files',
                 'INPUT_FILE_DESCRIPTION' => [ '.+$' ],
                 'INPUT_DIR_DEPTH' => 0,
                 'DO_NOT_DELETE'=> 0,

                 'JUNK_MATCH'   => [ 'TIME.*PAGE', 'AF-' ],
                 'BAD_CHARS'    => [ "\000", "\011", "\015" ],
                 'BLANK_MATCH'  => '^[ \t]*$',
                 'SUB_SPACES'   => '_',
                 'REPORT_END'   => '^\s*END',

                 'REPORT_TYPES' => [ ... ]
      }
```

3.1.1 Report-type specific configuration

An example list of report-specific configurations is shown here, followed by a description of each option.

```
      'REPORT_TYPES' => [
            {     TYPE_NAME    => 'STS',
                  LEADING_TEXT => '^\s*STATISTIC DATA REPORT',
                  LAYOUT       => 'multi-object',
                  REPORT_START => '\s*REPORT DATA',
                  OBJECT_START => '^\s*OBJTYPE',
                  FAILURE_HDR  => [ 'CFAIL' ],
                  FAILURE_OBJ  => [ '\s*FAULT CODE' ],
                  FAILURE_TXT  => [ '\s*FAULT INTERRUPT' ],
                  USE_NAMES    => 0,
                  TIME_TO_USE  => 'STIME',
                  EXTRACT_NODE => [ '(\w+).*', '$1' ]
            },
            {     TYPE_NAME    => 'CCITT7_1',
                  LEADING_TEXT => '^\s*CCITT7',
                  KEY_MATCH    => [ 'DEST', 'SPID' ],
                  LAYOUT       => 'tabular',
                  FAILURE_HDR  => [ 'FCODE' ]
            },
            }     TYPE_NAME    => 'CCITT7_2',
                  LEADING_TEXT => '^\s*CCITT7',
                  KEY_MATCH    => [ 'LS', 'SPID' ],
                  LAYOUT       => '2-level',
                  FAILURE_HDR  => [ 'FCODE' ]
            },
            }     TYPE_NAME    => 'TRARMET',
                  LEADING_TEXT => '\s*TRAFFIC MEASUREMENT',
                  LAYOUT       => 'tabular',
                  FAILURE_HDR  => [ 'FCODE' ]
            },
            }     TYPE_NAME    => 'TRARTMET',
                  LEADING_TEXT => '\s*TRAFFIC TYPE MEASUREMENT',
                  LAYOUT       => 'tabular',
                  FILLIN       => 'TRAFFTYPE',
                  FAILURE_HDR  => [ 'FCODE' ]
            },
            }     TYPE_NAME    => 'TRDIPMET',
                  LEADING_TEXT => '\s*TRAFFIC DISPERSION',
                  LAYOUT       => 'tabular',
                  PROPAGATE    => [ 'R', 'TRDG' ],
                  FAILURE_HDR  => [ 'FCODE' ]
            }
      ]
```

All report types must have at least the following fields set up: TYPE_NAME, LEADING_TEXT, LAYOUT and FAILURE_HDR. All others are optional and some are specific only to reports that have a certain LAYOUT set.

TYPE_NAME	The name of the report type will be reported in the Gateway log file. For report types with LAYOUT set to either 'tabular' or '2-level' the TYPE_NAME will form the first part of the output PIF name. (required)
LEADING_TEXT	A regular expression used to find the start of a report in the input. If more than one report type has LEADING_TEXT that matches the start of a report only the first will be used. (required)
LAYOUT	This tells the Gateway how the report will be laid out. It can take one of the following values: 'tabular', '2-level' and 'multi-object'
FAILURE_HDR	A list of keys to check in a report header. If any of these keys are set in the header the report will be considered to contain errors and will be logged as such and ignored by the Gateway, no PIF will be produced.

There are further optional rules that are common to all report layouts.

KEY_MATCH	This is a list of counter names which will be checked before the Gateway decides which type a report is. For example you might have two variants of CCITT7 report which, for example, you would configure two report types to have KEY_MATCH lists of ['DEST', 'SPID'] and ['LS', 'SPID'] respectively and configure them to have different settings. (optional)
TIME_TO_USE	Allows you to specifiy which field of a report's header contains the time that should be placed into the output PIF name. If no setting is given the Gateway will default to use 'TIME'. For example, an STS report has both STIME and ETIME (start and end times respectively) but not TIME in its header. You must decide which of these to use and configure the Gateway accordingly. (optional)
EXTRACT_NODE	A 2-part list which contains the left and right sides of a perl s/// operation. This substitution is performed on the source file name to get the node id which will be placed in the PIF header block NODEID column. The node id will also make up the part of the output PIF file name. If this option is not set the entire file name

	will be used as the node id. The example given above will take the text at the start of the source file name up to and excluding the first non-alphanumeric character as the node id. (optional)

3.1.2 Layout specific configuration

Multi-object reports

REPORT_START	A regular expression used to match the text present at the start of a multi-object report. This text is not mandatory in a report but you need to configure it here so the Gateway doesn't see the text as spurious input. If this text is not present in the multi-object report the Gateway will operate normally. (optional)
OBJECT_START	A regular expression matching the text present at the start of an object. (required)
FAILURE_OBJ	A regular expression used to match error text present on the same line as OBJECT_START is matched. If the match is made the object will be reported as bad and not written to PIF. (optional)
FAILURE_TXT	A regular expression used to match the first line of text in an object to check if the object contains error text instead of counter values. If the match is made the object will be reported as bad and not written to the PIF file. (optional)
USE_NAMES	A boolean option, where a True value: 1 or 'Yes', switches the option on, and a False value: 0 or '', will turn it off. If no value is given False is the default. If this option is switched on the text preceding the OBJECT_START match will be used as an object's type rather than the more usual behaviour of using the object type text following OBJECT_START. If this option is switched on and the report does not contain any text before the OBJECT_START the names will be set to 'anon' by default. (optional)

Tabular reports

FILLIN	Allows you to add a counter name to the start of a report. This option should only be used in special cases such as certain report types that do not include a counter name for the first column. (optional) For example:

	ORG	IEX	TE	OEX	
	TRAFF	253.1	376.6	0.0	624.6
	NCALLS	17703	26576	0	43047
	ICONG	0.0	0.0	0.0	0.0

PROPAGATE	A list of counter names that will be propagated as the report is parsed. This means that if a counter in this list has no value specified in the current line of the report the previous value will be repeated. (optional) For example:

	R	TRDG	TRD	TRAFF
	INT1I	0	0.0	
		2	33	0.0
	34	0.0		
	INT2I	0	0	0.0

		2	33	0.0
			34	0.0
		3	33	0.0
			34	0.0

When configured to have PROPAGATE => ['R', 'TRDG'] would be equivalent to:

	R	TRDG	TRD	TRAFF
	INT1I		0	0.0
	INT1I	2	33	0.0
	INT1I	2	34	0.0
	INT2I	0	0	0.0
	INT2I	2	33	0.0
	INT2I	2	34	0.0
	INT2I	3	33	0.0
	INT2I	3	34	0.0

3.2 C7PM Engine Rule

This Engine Rule allows you to process Ericsson C7PM files.

This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory	'R2P_C7PM' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of datathat are written to the data file.
DATA_BLOCKSIZE	mandatory	The size in bytes of the records contained within the block.

Example:

```
{
        'RULE_TYPE' => 'R2P_C7PM',
        'RULE_DESC' => 'C7PM Files',
        'INPUT_FILE_DESCRIPTION' => [ '.*C7PM.*$' ],

        'NUMBER_OF_FILES_TO_PROCESS' => 5,

        'FILENAME_HEADER_FIELDS' => {
                    'NODEID' => '^(\w+)-',
                    'FILENAME' => '^(.*)',
                    },
        'BLOCKSIZE' => 2048,
        'DATA_BLOCKSIZE'  => 96,
        'SKIP_FIRST_BYTES' => 0,
            'VI_CONFIG' => \%C7PM_FIELD_LIST,
    },
```

The following points should be noted about the example shown above:

- Only files containing the string C7PM in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array C7PM_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.

The naming of the output PIFS have format:

```
<NODEID>-#-<INPUT FILENAME>-#-<START_DATE>-#-<START_TIME>-#-
<INDEX>-#-<BLOCKNAME>-#-I.pif
```

It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<BLOCKNAME> in this case will either be 'DEST', 'LSCOUNT', 'SLCOUNT'
<INDEX> is an integer 0,1,2.. to ensure unique filenames.

3.3 MTCES Engine Rule

This Engine Rule allows you to process Ericsson MTCES and MCS files.

MTCES/MSC format data files both have two very similar formats. The format is made up of two types of records, A and B. Record A contains all the names of the counters reported in Record B. In one variant of the format Record A is only produced by the switch once a day. In the second variant Record A is produced at the start of each file.

This Engine Rule allows you to store the counter names from the Record A so that they are available for the Gateway to use when processing the next group of datafiles.

This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory 'R2P_MTCES' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory The size in bytes of the chunks of data that are written to the data file.
STORE_RECA	optional A pif is produced which contains details of the

	data contained within RECORD A

Example:

```
{
        'RULE_TYPE' => 'R2P_MTCES',
        'RULE_DESC' => 'MTCES Files',
        'INPUT_FILE_DESCRIPTION' => [ '.*MTCES.*$' ],

        'NUMBER_OF_FILES_TO_PROCESS' => 5,

        'FILENAME_HEADER_FIELDS' => {
                    'NODEID' => '^(\w+)-',
                    'FILENAME' => '^(.*)',
                    },
        'BLOCKSIZE' => 2048,
        'SKIP_FIRST_BYTES' => 0,
        'VI_CONFIG' => \%MTCES_AMPS_FIELD_LIST,
'STORE_RECA' => 'True',
},
```

The following points should be noted about the example shown above:

- Only files containing the string MTCES in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array MTCES_AMPS_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.

NOTE:
Two different versions (AMPS and TACS) of the format of MTCES file have been observed, therefore the example configuration EngineConfig.pm contains to associative arrays (hashes) MTCES_AMPS_FIELD_LIST and MTCES_TACS_FIELD_LIST. The former (AMPS) has in sub record 32 format extra data for START_YEAR, START_MONTH, START_DAY.

The naming of the output PIFS have format:

```
MTCES-#-<SELECTION_CRITERIA>-#-<EXCH_DESIGNATION>-#-
<NO_SUBRECORDS>-#-<NODEID>-#-<START_DATE>-#-<START_TIME>-#-I.pif
```

It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<SELECTION_CRITERIA> are one of the strings 'EXCH' or 'CELL'

3.4 MTRRCR Engine Rule

This Engine Rule allows you to process Ericsson MTRRCR files.
This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory	'R2P_MTRRCR' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of data that are written to the data file.

Example:

```
{
        'RULE_TYPE' => 'R2P_MTRRCR',
        'RULE_DESC' => 'MTRRCR Files',
        'INPUT_FILE_DESCRIPTION' => [ '.*MTRRCR.*$' ],

        'NUMBER_OF_FILES_TO_PROCESS' => 5,

        'FILENAME_HEADER_FIELDS' => {
                'NODEID' => '^(\w+)-',
                'FILENAME' => '^(.*)',
                },
        'BLOCKSIZE' => 2048,
        'SKIP_FIRST_BYTES' => 0,
        'VI_CONFIG' => \%MTRRCR_FIELD_LIST,
},
```

The following points should be noted about the example shown above:

- Only files containing the string MTRRCR in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array MTRRCR_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.

The naming of the output PIFS have format:

```
<INPUT FILENAME>-#-<NODEID>-#-<START_DATE>-#-<START_TIME>-#-
<INDEX>-#-I.pif
```

It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<INDEX> is an integer 0,1,2… to ensure unique filenames

3.5 MTRVC Engine Rule

This Engine Rule allows you to process Ericsson MTRVC files.
This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory	'R2P_MTRVC' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of data that are written to the data file.
DATA_BLOCKSIZE	mandatory	The size in bytes of the records contained within the block.

Example:

```
{
        'RULE_TYPE' => 'MTRVC',
        'RULE_DESC' => 'MTRVC Files',
        'INPUT_FILE_DESCRIPTION' => [ '.*MTRVC.*$' ],

        'NUMBER_OF_FILES_TO_PROCESS' => 5,

        'FILENAME_HEADER_FIELDS' => {
                    'NODEID' => '^(\w+)-',
                    'FILENAME' => '^(.*)',
                    },
        'BLOCKSIZE' => 2048,
        'DATA_BLOCKSIZE'  => 96,
        'SKIP_FIRST_BYTES' => 0,
        'VI_CONFIG' => \%MTRVC_FIELD_LIST,
},
```

The following points should be noted about the example shown above:

- Only files containing the string MTRVC in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array MTRVC_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.

The naming of the output PIFS have format:

```
<INPUT FILENAME>-#-<NODEID>-#-<START_DATE>-#-<START_TIME>-#-
<TYPE>-#-<INDEX>-#-I.pif
```

It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<TYPE> record type, 1,..7
<INDEX> is an integer 0,1,2.. to ensure unique filenames

3.6 SEQS Engine Rule

This Engine Rule allows you to process Ericsson SEQS files.

This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory	'R2P_SEQS' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of data that are written to the data file.

Example:

```
        {
                'RULE_TYPE' => 'R2P_SEQS',
                'RULE_DESC' => 'SEQS Files',
                'INPUT_FILE_DESCRIPTION' => [ '.*SEQS.*$' ],

                'NUMBER_OF_FILES_TO_PROCESS' => 5,

                'FILENAME_HEADER_FIELDS' => {
                            'NODEID' => '^(\w+)-',
                            'FILENAME' => '^(.*)',
                            },
                'BLOCKSIZE' => 2048,
                'SKIP_FIRST_BYTES' => 0,
                'VI_CONFIG' => \%SEQS_FIELD_LIST,
},
```

The following points should be noted about the example shown above:

- Only files containing the string SEQS in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array SEQS_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.

The naming of the output PIFS have format:

```
<INPUT FILENAME>-#-<NODEID>-#-<START_DATE>-#-<START_TIME>-#-
<INDEX>-#-I.pif
```

It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<INDEX> is an integer 0,1,2.. to ensure unique filenames

3.7 STFIOP Engine Rule

This Engine Rule allows you to process Ericsson STFIOP files.
This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory	'R2P_STFIOP' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of data that are written to the data file.
GEN_REC_SIZE	mandatory	The size in bytes of the GDR record

Example:

```
{
        'RULE_TYPE' => 'R2P_STFIOP',
        'RULE_DESC' => 'STFIOP Files',
        'INPUT_FILE_DESCRIPTION' => [ '.*STFIOP.*$' ],

        'NUMBER_OF_FILES_TO_PROCESS' => 5,

        'FILENAME_HEADER_FIELDS' => {
                    'NODEID' => '^(\w+)-',
                    'FILENAME' => '^(.*)',
                    },
        'BLOCKSIZE' => 2048,
        'GEN_REC_SIZE'  => 78,
        'SKIP_FIRST_BYTES' => 0,
        'VI_CONFIG' => \%STFIOP_FIELD_LIST,
},
```

The following points should be noted about the example shown above:

- Only files containing the string STFIOP in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array STFIOP_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.

The naming of the output PIFS have format:

```
<OBJ_TYPE_ID>-#-<NODEID>-#-<START_DATE>-#-<START_TIME>-#-<INDEX>-
#-I.pif
```

<OBJ_TYPE_ID> is the STS object type.

It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<INDEX> is an integer 0,1,2.. to ensure unique filenames.

3.8 TRAR Engine Rule

This Engine Rule allows you to process Ericsson TRAR files.
This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory	'R2P_TRAR' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of data that are written to the data file.
ADM_RECORD_SIZE	mandatory	The size in bytes of the ADM record
RECORD_SIZE	mandatory	The size in bytes of the other valid data records

Example for r9.1:

```
{
        'RULE_TYPE' => 'R2P_TRAR',
        'RULE_DESC' => 'TRAR Files for 9.1',
        'INPUT_FILE_DESCRIPTION' => [ '.*TRAR.*$' ],

        'NUMBER_OF_FILES_TO_PROCESS' => 5,

        'FILENAME_HEADER_FIELDS' => {
                    'NODEID' => '^(\w+)-',
                    'FILENAME' => '^(.*)',
                    },
        'BLOCKSIZE' => 2048,
        'ADM_RECORD_SIZE' => 71,
        'RECORD_SIZE' => 80,
        'SKIP_FIRST_BYTES' => 0,
        'VI_CONFIG' => \%TRAR_FIELD_LIST,
},
```

The following points should be noted about the example shown above:

- Only files containing the string TRAR in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array TRAR_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, their position in the data blocks and their lengths.

The naming of the output PIFS have format:

```
<OBJECT>-#-<NODEID>-#-<INPUT FILENAME>-#-<START_DATE>-#-
<START_TIME>-#-<INDEX>-#-I.pif
```

<OBJECT> i.e. ROUTELOSS, ROUTEQUEUE, ROUTEPBX
It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<INDEX> is an integer 0,1,2.. to ensure unique filenames

3.9 TRART Engine Rule

This Engine Rule allows you to process Ericsson TRART files.
This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	Mandatory	'R2P_TRART' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of data that are written to the data file.
ADM_RECORD_SIZE	mandatory	The size in bytes of the ADM record
DEVDATA_RECORD_SIZE	mandatory	The size in bytes of the DEVDATA record
DEVDATA_PER_BLOCK	mandatory	The number of DEVDATA records expected in a block
TRAFFCNT_RECORD_SIZE	mandatory	The size in bytes of the TRAFFCNT record
TRAFFCNT_PER_BLOCK	mandatory	The number of TRAFFCNT records expected in a block
TRAFF_TYPE	mandatory	A reference to a hash containing the traff_types for DEVDATA record

Example for r9.1:

```
{
        'RULE_TYPE' => 'R2P_TRART',
        'RULE_DESC' => 'TRART Files for r9.1',
        'INPUT_FILE_DESCRIPTION' => [ '.*TRART.*$' ],

        'NUMBER_OF_FILES_TO_PROCESS' => 5,

        'FILENAME_HEADER_FIELDS' => {
                    'NODEID' => '^(\w+)-',
                    'FILENAME' => '^(.*)',
                    },
        'ADM_RECORD_SIZE' => 32,
        'BLOCKSIZE' => 960,
        'DEVDATA_RECORD_SIZE' => 64,
        'DEVDATA_PER_BLOCK' => 4,
        'TRAFFCNT_RECORD_SIZE' => 86,
        'TRAFFCNT_PER_BLOCK' => 14,
        'SKIP_FIRST_BYTES' => 0,
        'VI_CONFIG' => \%TRART_FIELD_LIST,
            'TRAFF_TYPE' => \%trafftype,
    },
```

The following points should be noted about the example shown above:

- Only files containing the string TRART in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array TRART_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.
- The DEVDATA record is 64 bytes and will have 4 records, these 4 records are described by the configuration in the %traff_type hash
- The TRAFFCNT record is 86 bytes and will have 14 records in each block.

The naming of the output PIFS have format:

```
    <OBJECT>-#-<NODEID>-#-<INPUT FILENAME>-#-<START_DATE>-#-
    <START_TIME>-#-<INDEX>-#-I.pif
```

<OBJECT> i.e. DEVDATA, TRAFFCNT
It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<INDEX> is an integer 0,1,2.. to ensure unique filenames

3.10 TRDIP Engine Rule

This Engine Rule allows you to process Ericsson TRDIP files.
This Engine Rule allows you to use all the standard options that are specified in Engine.README. In addition, the configuration options specific to this Engine Rule are:

RULE_TYPE	mandatory	'R2P_TRDIP' - The string the Gateway uses to identify the type of this rule.
VI_CONFIG	mandatory	A reference to a hash containing the files format specification.
SKIP_FIRST_BYTES	optional	The number of bytes that should be skipped at the start of the record.
BLOCKSIZE	mandatory	The size in bytes of the chunks of data that are written to the data file.
ADM_RECORD_SIZE	mandatory	The size in bytes of the ADM record
RECORD_SIZE	mandatory	The size in bytes of the other valid data records

Example:

```
    {
            'RULE_TYPE' => 'R2P_TRDIP',
            'RULE_DESC' => 'TRDIP Files',
            'INPUT_FILE_DESCRIPTION' => [ '.*TRDIP.*$' ],

            'NUMBER_OF_FILES_TO_PROCESS' => 5,
            'FILENAME_HEADER_FIELDS' => {
                        'NODEID' => '^(\w+)-',
                        'FILENAME' => '^(.*)',
                        },
            'BLOCKSIZE' => 2048,
            'ADM_RECORD_SIZE' => 64,
            'RECORD_SIZE' => 64,
            'SKIP_FIRST_BYTES' => 0,
            'VI_CONFIG' => \%TRDIP_FIELD_LIST,
      },
```

The following points should be noted about the example shown above:

- Only files containing the string TRDIP in the name will be processed.
- All the characters before the first '-' in the filename will be inserted into the header block associated with the field 'NODEID'.
- The whole filename will be inserted into the header block associated with the field 'FILENAME'.
- The associative array TRDIP_FIELD_LIST contains the information that describes the file format. This associative array contains the counter names, there position in the data blocks and there lengths.

The naming of the output PIFS have format:

```
      <NODEID>-#-<INPUT FILENAME>-#-<START_DATE>-#-<START_TIME>-#-
      <INDEX>-#-I.pif
```

It assumes that there will be a key NODEID within the FILENAME_HEADER_FIELDS or DIRECTORY_HEADER_FIELDS configuration.
<INDEX> is an integer 0,1,2.. to ensure unique filenames

4 Post Parser Rules and Configuration

This section describes specific configurations needed to for the post parsing rules specific to the Ericsson GSM parser.

4.1 FILES_TO_KEEP

This rule removes a file after a specified period of time.
The configuration options available for this module are as follows:

INPUT_FILE_DESCRIPTION	a regular expression describing the type of file that this rule should be applied to.
HOURS_TO_WAIT	the number of hours to wait before a file is removed.
PRODUCE_PIF = "True"	will produce the output as a PIF file
= 0	will not produce a PIF file
PRODUCE_LIF = "True"	will produce the output as a LIF file
= 0	will not produce a LIF file

Example:

```
{       'RULE_TYPE' => 'FILES_TO_KEEP',
        'INPUT_FILE_DESCRIPTION' => ["^MTCES_RECORD_A-#-"],
        'HOURS_TO_WAIT'=> 24,
        'PRODUCE_PIF' => 0,
        'PRODUCE_LIF' => 0,
},
```

In this example, the file which contains the expression in INPUT_FILE_DESCRIPTION is deleted after 24 hours of its creation time.

4.2 TRANSPOSE_TABLE

The Transpose Table post-parser tool takes a single column of counter values and rearranges a PIF block as shown in the example below.
The following configuration options are available for the TRANSPOSE_TABLE post-parser tool:

INPUT_FILE_DESCRIPTION	a list of regular expressions used to match the names of files to be processed by this tool
OUTPUT_FILENAME_START	text that is to be prepended to the name of files output by this tool, in addition to adding '-#-CM' to the name (optional)
OUTPUT_BLOCK_NAME	if this option is set then the block names in the input PIF will be ignored and all blocks output by this tool will use this name (optional)
REDUNDANT_COUNTERS	a list of counter names to remove from the input (optional)
PRODUCE_PIF	if set then a PIF will be created in the intermediate directory (optional)
PRODUCE_LIF	if set then a LIF will be created in the output directory (optional)
TRANSPOSE_ON	the name of the counter that the table wll be transposed about (required)

For example:

```
{       'RULE_TYPE'                  => 'TRANSPOSE_TABLE',
        'RULE_DESC'              => 'Transpose data in PIF block',
        'INPUT_FILE_DESCRIPTION' => ['TRARTMET.*-#-I.pif'],
        'REDUNDANT_COUNTERS'     => [ 'IEXTE' ],
        'PRODUCE_PIF'            => 'True',
        'PRODUCE_LIF'            => 'True',
        'TRANSPOSE_ON'           => 'TRAFFTYPE'
}
```

Using the above configuration the following transposition would occur. The appearance of the PIF has been formatted to increase readability.

ORG	TE	TRAFFTYPE	IEX
0.0	0.0	icong	0.0
253.1	0.0	traff	376.6
0.0		econg	0.0
17703	0	ncalls	26576
6.4	0.0	unsuc	2.9

becomes:

icong	traff	econg	ncalls	TRAFFTYPE	unsuc
0.0	253.1	0.0	17703	ORG	6.4
0.0	0.0		0	TE	0.0
0.0	376.6	0.0	26576	IEX	2.9

4.3 STRING_SPLIT

This allows a counter values to be split in a desired way.There are 2 methods
1) The counter value can be split in two by specifing a delimiter (STRING_DELIMITER, (default value '-') and list a list of counter names in COUNTERS_TO_SPLIT whose counter value you wish to split

i.e. obj_id MOBILE-INCOME would be converted into:

```
obj_id_1 MOBILE
obj_id_2 INCOME
```

where
```
'STRING_DELIMITER' => '-',
'COUNTERS_TO_SPLIT' => ['obj_id'],
```

2) The counter value can be split into any number of parts by speficying regular expressions for each counter name in ENHANCED_COUNTER_SPLIT option.

i.e. obj_id MOBILE-1:INCOME-2 would be converted into

```
obj_id_1 MOBILE
obj_id_2 1
obj_id_3 INCOME
obj_id_4 2
```

where
```
ENHANCED_COUNTER_SPLIT => {
        'obj_id' => [ '(\w+)-(\d):(\w+)-(\d)'],
}
```

The configuration options available for this module are as follows:

INPUT_FILE_DESCRIPTION	a regular expression describing the type of file that this rule should be applied to.
OUTPUT_BLOCK_NAME	The name that should be used for the section name in the loader file.
PRODUCE_PIF = "True"	will produce the output as a PIF file
= 0	will not produce a PIF file
PRODUCE_LIF = "True"	will produce the output as a LIF file
= 0	will not produce a LIF file
STRING_DELIMITER	(default value '-') this is the delimiter used to identify where the string is to be split.
COUNTERS_TO_SPLIT	This is the list of counters that will be split into two strings. The new strings are named <counter_name>_1 and <counter_name>_2 (where counter_name is the original name of the counter being split.
ENHANCED_COUNTER_SPLIT	This is a hash where the hash keys are counter names that wish to be split in some way. The hashes elements reference a list of regular expressions which will try to split the relevant counter name value in this way. The regular expressions in the list should be ordered in order of most preferable first. The new strings are named <counter_name>_1, <counter_name>_2 etc.(where counter_name is the original name of the counter being split.
REDUNDANT_COUNTERS	a list of counters that should be deleted from the resultant output.

Example:

```
{      'RULE_TYPE' => 'STRING_SPLIT',
        'RULE_NAME' => 'Split obj_id into two',
        'INPUT_FILE_DESCRIPTION' => [ "NCELLRXT-#-.*-#-I.pif"],
        'PRODUCE_PIF' => "True",
        'PRODUCE_LIF' => 0,
        'ENHANCED_COUNTER_SPLIT' => {
             'OBJ_ID'  => ['(\w+)[-\s](\w+)',],
             'COUNTER_A' => ['(\w{3}):(\w{3}):(\w+)',
'(\w{3}):(\w{1})(\w+)'],
       },
        'OUTPUT_BLOCK_NAME' => "ANYTHING",
        'REDUNDANT_COUNTERS' => ["NO_COUNTERS", "NO_OBJ_RECORDS"],
},
```

In this example, the counter OBJ_ID is to be split into two, where the delimiter is '-' or ' '. The counter COUNTER_A will be split into 3 parts, if it cannot make a match of the value with the first regular expression, then the second will be tried, e.g.

COUNTER_A = 'abc:def:g'	Use 1st RED ->	**`COUNTER_A_1 = 'abc', COUNTER_A_2 = 'def', COUNTER_A_3 = 'g'`**
COUNTER_A = 'abc:defg'	1st RE fails use 2nd RE ->	**`COUNTER_A_1 = 'abc', COUNTER_A_2 = 'd', COUNTER=_A_3 = 'efg'`**

Note: if you have a different number of tagged expressions within the regular expressions for the list for a particular counter name, then you will run the danger of not all the subsections of the counter value being split as intended.

A PIF will be produced, a LIF will not. The outputted blocknames will be called ANYTHING. NO_COUNTERS and NO_OBJ_RECORDS will be excluded from the outputted file (these are REDUNDANT_COUNTERS).
Files processed by STRING_SPLIT have a filename which ends with a -#-E.pif

4.4 NAME_BLOCK_CAT

Basically this rule joins all the blocks into one block, and can add a counter to the start of other counter names.
This takes a counter (i.e. OBJ_ID) and places it in front of the counters belonging to this OBJ_ID, and then removes the old names, i.e.:

Orig:	OBJ_ID CANCLOC		New:	CANCLOC_NUMTOT 344
	NUMTOT 344			CANCLOC_NUMSUC 208
	NUMSUC 208			

It also puts all the blocks in the file into the one block:

Orig:	HLRMAP{		New:	HLRMAP{
	blah 123			blah 123
	}			popp 432
	HLRMAP{			}
	popp 432			
	}			

The configuration options available for this module are as follows:

INPUT_FILE_DESCRIPTION	a regular expression describing the type of file that this rule should be applied to.
OUTPUT_BLOCK_NAME	The name that should be used for the section name in the loader file.
PRODUCE_PIF = "True"	will produce the output as a PIF file
= 0	will not produce a PIF file
PRODUCE_LIF = "True"	will produce the output as a LIF file
= 0	will not produce a LIF file
LEADING_COUNTER	This is the counter which will precede thecounter names. Its default is 'OBJ_ID'.
FILENAME_ADDITION	This also the addition of a word in the output filename: PLMN-#-ASRF220-#-<word>-#-.pif Where <word> is defined here.

Example:

```
{      'RULE_TYPE' => 'NAME_BLOCK_CAT',
        'RULE_NAME' => 'Cat obj_id to counters',
        'INPUT_FILE_DESCRIPTION' => [ "HLRMAP.*-#-I.pif"],
        'LEADING_COUNTER' => 'OBJ_ID',
        'REDUNDANT_COUNTERS' => ["NO_COUNTERS", "OBJ_TYPE_ID",
"OBJ_GROUP_NO"],
        'PRODUCE_PIF' => "True",
        'PRODUCE_LIF' => "True",
        'FILENAME_ADDITION' => "CAT",
},
```

Files processed by NAME_BLOCK_CAT have a filename which ends with a -#-EH.pif

4.5 SPLIT_IO

This rule is designed to assist in recognising route direction (i.e. incoming or outgoing). Once the block is identified as to its direction, the counters within that block are prefixed or suffixed with the direction code (user defined (normally I or O), and all recognised blocks are outputted to another file.
The configuration options available for this module are as follows:

INPUT_FILE_DESCRIPTION	a regular expression describing the type of file that this rule should be applied to.
OUTPUT_BLOCK_NAME	The name that should be used for the section name in the loader file.
PRODUCE_PIF = "True"	will produce the output as a PIF file
= 0	will not produce a PIF file
COUNTER_TO_DETERMINE_IO	this is the counter that will be used to determine if this blocks is required. The last character of this counter must match either the PREFIX_TO_USE or SUFFIX_TO_USE (depending on which one is used). If it does not match this, this block will not be used in the outputted file. The last character of this counter will be removed.
PREFIX_TO_USE or	The character to search on. This will also be prefixed (with an underscore) to all the counter names in that block (unless stated otherwise).
SUFFIX_TO_USE	The character to search on. This will also be suffixed (with an underscore) to all the counter names in that block (unless stated otherwise).
COUNTER_NAMES_NOT_TO_CHANGE	a list of counters who's name's will not change.
REDUNDANT_COUNTERS	a list of counters that should be deleted from the resultant output.

Example:

```
{       'RULE_TYPE' => 'SPLIT_IO',
        'COUNTER_TO_DETERMINE_IO' => "ROUTE_NAME",
        'PREFIX_TO_USE' => "I",
        'INPUT_FILE_DESCRIPTION' => [ "TRAR.*-#-I.pif"],
        'REDUNDANT_COUNTERS' => ["FILLER"],
        'COUNTER_NAMES_NOT_TO_CHANGE' => [ "NO_DEVICES", "SUM_BDEVS" ],
        'PRODUCE_PIF' => "True",
},
```

In this example, the rule SPLIT_IO uses the ROUTE_NAME to determine the direction of travel. In order to establish the direction to search on the PREFIX_TO_USE is set as 'I', in this case if the ROUTENAME ends with the character 'I' it will be processed (i.e. a ROUTENAME GISUWI is incoming, and any other ending will be ignored). The PREFIX 'I' plus a '_' will be prefixed to all the counters in that block except the

COUNTER_TO_DETERMINE_IO and REDUNDANT_COUNTERS (which will be removed from the output file) and the COUNTER_NAMES_NOT_TO_CHANGE. For example:

** BEFORE **		** AFTER **
ROUTELOSS{		**ROUTELOSS{**
NO_BIDS 007		**I_NO_BIDS 007**
SUM_BDEVS 019		**SUM_BDEVS 019**
NO_BANSWERS 02469		**I_NO_BANSWERS 02469**
ROUTE_NAME DCTAI		**ROUTE_NAME DCTA**
NO_PLOPS 0169		**I_NO_PLOPS 0169**
NO_DEVICES 0362		**NO_DEVICES 0362**
FILLER		**}**
}		

Files processed by SPLIT_IO have a filename which ends with a:
-#-SP<suffix/prefix>.pif (where suffix or prefix are defined in the rule).

4.6 CHOP_COUNTER

This rule chops a number of characters off the end of a specified counter.
The configuration options available for this module are as follows:

INPUT_FILE_DESCRIPTION	a regular expression describing the type of file that this rule should be applied to.
OUTPUT_BLOCK_NAME	The name that should be used for the section name in the loader file and as an extension for all output files.
PRODUCE_PIF = "True"	will produce the output as a PIF file
= 0	will not produce a PIF file
PRODUCE_LIF = "True"	will produce the output as a LIF file
= 0	will not produce a PIF file
COUNTER_TO_CHOP	The counter which is to be chopped
COUNTER_NAME_FOR_CHOPPED_COUNTER	The name of the NEW counter produced from the chopped counter.
NO_CHARS_TO_CHOP_OFF_END	The number of characters to remove from the end of the counter name.
REDUNDANT_COUNTERS	a list of counters that should be deleted from the resultant output.

Example:

```
{       'RULE_TYPE' => 'CHOP_COUNTER',
        'INPUT_FILE_DESCRIPTION' => [ ".*-#-I.pif"],
        'COUNTER_TO_CHOP' => "ROUTE_NAME",
        'COUNTER_NAME_FOR_CHOPPED_COUNTER' => "RT",
        'NO_CHARS_TO_CHOP_OFF_END' => '3',
        'PRODUCE_PIF' => "True",
        'PRODUCE_LIF' => "True",
}
```

In this example, the counter to be chopped is ROUTE_NAME. 3 Characters are to be chopped off the end of the value, and the result placed into the counter RT.
Files processed by CHOP_COUNTER have a filename which ends with a -#-CC.pif

4.7 ROUTE_DIRECTION

This rule obtains the route direction, then adds this to the output file. This is very specific, with ROUTELOSS files in mind.
The technique used to do this has been updated in r2.1.3 to better manage asynchronous routes (bug 26849). This it does by applying the following rules;

- The route is "bothway" if incoming circuits == outgoing circuits
- If incoming circuits <> outgoing circuits
    - By reading in the counter name which would hold incoming values it can determine if the direction of travel is OUTGOING (if the incoming value is null.
    - By reading in the counter name which would hold outgoing values it can determine if the direction of travel is INCOMING (if the outgoing value is null.
    - If neither of these counters are null, then the route is "asynchronous bothway". This will need special handling to write separate records for input and output counters.

The configuration options available for this module are as follows:

INPUT_FILE_DESCRIPTION	a regular expression describing the type of file that this rule should be applied to.
OUTPUT_BLOCK_NAME	The name that should be used for the section name in the loader file and as an extension for all output files.
PRODUCE_PIF = "True"	will produce the output as a PIF file
= 0	will not produce a PIF file
PRODUCE_LIF = "True"	will produce the output as a LIF file
= 0	will not produce a PIF file
REDUNDANT_COUNTERS	a list of counters that should be deleted from the resultant output.
COUNTERS_TO_SEARCH_ON_FOR_IN	the counters that should hold INCOMING values.
COUNTERS_TO_SEARCH_ON_FOR_OUT	the counters that should hold OUTGOING values.
CODE_FOR_INCOMING	the outputted code for this direction
CODE_FOR_OUTGOING	the outputted code for this direction
CODE_FOR_BOTH_DIRECTIONS	the outputted code for this direction
OUTPUT_COUNTER_NAME	the counter name which will be added to the output file, containing
INCOMING_CIRCUIT_NO_COUNTER	The number of incoming circuits or devices.
OUTGOING_CIRCUIT_NO_COUNTER	The number of outgoing circuits or devices.
ROUTE_NAME_COUNTER	Counter containing route name.

Example:

```
{      'RULE_TYPE' => 'ROUTE_DIRECTION',
       'INPUT_FILE_DESCRIPTION'  => ["ROUTE.*J.pif"],
       'COUNTER_TO_SEARCH_ON_FOR_IN' => "NO_UBIDS_I",
       'COUNTER_TO_SEARCH_ON_FOR_OUT' => "NO_UBIDS_O",
       'CODE_FOR_INCOMING'            => 'I',
       'CODE_FOR_OUTGOING'            => 'O',
       'CODE_FOR_BOTH_DIRECTIONS'     => 'B',
       'OUTPUT_COUNTER_NAME'          => "ROUTE_DIR",
        'REDUNDANT_COUNTERS'           => [],
        'OUTPUT_BLOCK_NAME'            => "",
        'ROUTE_NAME_COUNTER'           => "ROUTE_NAME",
        'INCOMING_CIRCUIT_NO_COUNTER' => "NO_DEVICES_I",
        'OUTGOING_CIRCUIT_NO_COUNTER' => "NO_DEVICES_O",
        'PRODUCE_LIF'                  => "True",
        'PRODUCE_PIF'                  => 0,
},
```

In this example,
In order to see if the file has incoming data, then the counter NO_UBIDS_I is searched on. If the value for this counter is NULL, then it can be assumed the direction is OUTGOING, so O (CODE_FOR_OUTGOING) is assigned to ROUTE_DIR (OUTPUT_COUNTER_NAME). If NO_UBIDS_O and NO_UBIDS_I both contain values, then the direction is both-ways, so CODE_FOR_BOTH_DIRECTIONS 'B' is assigned.
The only output file produced in this example is a LIF.
Files processed by ROUTE_DIRECTION have a filename which ends with a -#-R.pif

5 Configuration Support

Configurations listed below are available for the systems releases from Ericsson GSM.

5.1 Sample Configuration

There are sample configurations used for Ericsson GSM Gateway within the examples directory for the following system releases:

- GSM/BSS V11

5.2 Tech Pack Support

Tech pack support is included in the Ericsson GSM Gateway for the following data version.

- GSM/BSS V9.1
- GSM/BSS V10
- GSM/BSS V11
- GSM/BSS V1206b

The EngineConfig.pm and UserConfig.pm configurations for these Tech Packs are located in the tech_pack_support kit sub directory.