General information on configuring the presence of SEIs:

- There are four lists associated with each SEI configuration, which can be set in a corresponding SEI cfg-file to specify pictures for which the SEI is send. The lists are:
  - ApplicableLayerIds
  - ApplicablePocs
  - ApplicableTids
  - ApplicableVclNaluTypes
- When a list is empty, it is handled as if it would include all possible values.
- There can be multiple cfg-files for the same SEI payload type, with different configurations.
- SEI cfg-files can be specified in the encoder cfg-file with the parameter `SeiCfgFileName_N`.
- Examples for SEI cfg-files are given in `/cfg/SEIs.`
- An SEI is inserted to the bitstream as specified in the following:
  - Let SeiX an SEI configuration with lists ApplicableLayerIdsX, ApplicablePocsX, ApplicableTidsX, and ApplicableVclNaluTypesX.
  - An SEI with configuration SeiX is inserted as leading SEI of picA, when picture picA has nuh_layer_id equal to nuhLayerIdA, TemporalId equal to TIdA, PicOrderCntVal equal to pocA, and nal_unit_type equal to naluUnitTypeA such that all of the following conditions are true:
    - nuhLayerIdA is an element of ApplicableLayerIdsX.
    - pocA is an element of ApplicablePocsX.
    - TIdA is an element of ApplicableTidsX.
    - naluUnitTypeA is an element of ApplicableVclNaluTypesX.

Required changes to enable new SEIs in the code:

There is already some automatically generated inactive code for the new SEIs in HTM. Following steps are necessary to enable it for an SEI with name `SEIName`:

Sei.h

- Change scope of `NH_MV_SEI_TBD` such that it no longer includes `class SEIName.`
- When you don't intend to setup/modify the SEI automatically by the encoder, but want to use an SEI cfg-file only, remove `setupFromSlice` in `class SEIName`.

Sei.cpp

- `SEIName::setupFromSlice`
  - If you intend to setup/modify the SEI automatically by the encoder, modify the `SEIName` members in this function using data from slice and change scope of `NH_MV_SEI_TBD` such that it no longer includes `SEIName::setupFromSlice.`
  - Otherwise, (you don't intend to setup/modify the SEI automatically by the encoder), remove `SEIName::setupFromSlice.`
- `SEIName::setupFromCfgFile`
  - Change scope of `NH_MV_SEI_TBD` such that it no longer includes `SEIName::setupFromCfgFile`.
  - Set default values for `defAppLayerIds, defAppPocs, defAppTids, defAppVclNaluTypes` in a way that the SEI is send with pictures that would be typically.
  - When a setup or modification of the SEI by the encoder is not indented, set `defModifyByEncoder` to `false`.
  - For member variables that are arrays change `ADDNUM` and to the maximum expected size of the respect array.
    - E.g. you have a 3D-Array `m_foo[x][y][z]` of maximum size (MAX_X * MAX_Y * MAX_Z), you should have `(Foo_%d_%d, m_foo, IntAry1d (`MAX_Z`,0),`MAX_X`,` MAX_Y`, Foo)`
    - For cfg-file parsing, this will expand to `Foo_x_y`, with `x` and `y` in the range of 0 to MAX_X and MAX_Y, respectively. For configuration each parameter `Foo_x_y` can have multiple space-separated entries (one for each z).

  - If default values for member variables don't comply with the spec, change them. Otherwise keep the zero initialization.
- `SEIName::checkCfg`
  - Add checks on constraints on presence of the SEI as in the spec.
  - Add checks on values of syntax elements as in the spec.
  - Remove unused lines.
- `SEI::getNewSEIMessage`
  - Change scope of `NH_MV_SEI_TBD` such that it no longer includes `case SEI::SEI_NAME : return new SEIName`;

SEIwriter.h

  - Change scope of `NH_MV_SEI_TBD` such that it no longer includes `xWriteSEIName`.

SEIwriter.cpp

- `SEIWriter::xWriteSEIName`
  - Change scope of `NH_MV_SEI_TBD` such that it no longer includes `SEIWriter::xWriteSEIName`
  - Modify code of `SEIWriter::xWriteSEIName` such that writing is possible, this may include:
    - Fixing syntax.
    - Implantation of `getSyntaxElementNameLen` functions providing the length of syntax elements.
    - In some cases data from a scalable nesting SEI associated with the SEI might be required. For this a pointer `m_scalNestSeiContThisSei` is provided in `class SEI.` When the SEI is not nested the pointer is equal to `NULL. (`This is currently the only possible value, but might change in future.)
- `SEIWriter::xWriteSEIpayloadData`
  - Change scope of NH_MV_SEI_TBD such that it no longer includes the `case SEI::SEI_NAME:` and related lines.

SEIread.h

- Change scope of `NH_MV_SEI_TBD` such that it no longer includes `xParseSEIname.`

SEIread.cpp

- `SEIReader::xParseSEIName`
  - Change scope of `NH_MV_SEI_TBD` such that it no longer includes `SEIReader::xParseSEIName`
  - Modify code of `SEIReader::xParseSEI( const SEIName& sei)` such that parsing is possible, this may include:
    - Fixing syntax
    - Resizing of arrays.
    - Reusing the `getSyntaxElementNameLen` functions providing the length of syntax elements.
- `SEIReader::xReadSEImessage`
  - Change scope of `NH_MV_SEI_TBD` such that it no longer includes the `case SEI::SEI_NAME:` and related lines.

/cfg/SEI/seiname.cfg

- Add the correct `PayloadType` value.
- Set some typical values for `ApplicableLayerIds`, `ApplicablePocs`, `ApplicableTids`, `ApplicableVclNaluTypes`
- Set some exemplary values for the payload data.
- If the configuration can be set by the encoder, set `ModfiyByEncoder` equal to 1. Otherwise, set `ModifyByEncoder` equal to 0.
- If necessary, expand parameters for arrays (e.g. add `Foo_0_1`, `Foo_0_2` ...).