Last updated: EPN, Thu Mar 17 10:45:20 2016

Sections of this file:
 (1) The v5 error codes
 (2) Error code information in the .all.errors output file
 (3) Error code information in the .peraccn.errors output file
 (4) Thoughts on 'really bad' error combinations that could be
     used to identify and filter problematic sequences
 (5) List of required error combinations
 (6) List of incompatible error combinations
 (7) Other known restrictions of error combinations
 (8) Notes on olp errors
 (9) Notes on stp, trc, ext and nst errors
(10) Miscellaneous notes

## (1) The v5 error codes

| idx | code | example-error-message |
|---|---|---|
| 1 | ori | there is not exactly 1 occurrence of origin sequence [0 occurrences] |
| 2 | nop | unable to identify homologous feature |
| 3 | nm3 | length of nucleotide feature is not a multiple of 3 [1900] |
| 4 | bd5 | alignment to reference does not extend to 5' boundary of reference [2 nt from 5' end] |
| 5 | bd3 | alignment to reference does not extend to 3' boundary of reference [7 nt from 3' end] |
| 6 | olp | feature does not overlap with same set of features as in reference [-(1.1,6.1)] |
| 7 | str | predicted CDS start position from homology search is not beginning of ATG start codon [AAA] |
| 8 | stp | predicted CDS stop  position from homology search is not end of valid stop codon (TAG\|TAA\|TGA) [AAT] |
| 9 | ajb | feature (MP or CDS) is not adjacent to same set of features before it as in reference [NONE != 14.1 (ref)] |
| 10 | aja | feature (MP or CDS) is not adjacent to same set of features after it as in reference [NONE != 14.1 (ref)] |
| 11 | trc | in-frame stop codon exists 5' of stop position predicted by homology to reference [homology search predicted 1801..1360 revised to 1801..1376 (stop shifted 16 nt)] |
| 12 | ext | first in-frame stop codon exists 3' of stop position predicted by homology to reference [homology search predicted 1801..1360 revised to 1801..1351 (stop shifted by 9 nt)] |
| 13 | ntr | mature peptide is not translated because its CDS has an in-frame stop 5' of the mature peptide's predicted start [stop at positions 1498-1500] |
| 14 | nst | no in-frame stop codon exists 3' of predicted valid start codon |
| 15 | aji | CDS comprised of mat_peptides has at least one adjacency inconsistency between 2 mat_peptides [MP#7 (2420..3475) not adjacent to MP#8 (3477..3506)] |
| 16 | int | CDS comprised of mat_peptides is incomplete: at least one mat_peptide is not translated due to early stop (ntr) [MP#8, MP#9, MP#10, MP#11, MP#12, MP#13, MP#14] |
| 17 | inp | CDS comprised of mat_peptides is incomplete: at least one mat_peptide is not identified (nop)  [MP#12] |

## (2) Error code information in the .all.errors output file

The annotation script (dnaorg_annotate_genomes.pl) produces an output
file with the suffix <s>.all.errors. This file includes a list of all
errors found, one error per line, with informative error messages such as
those present in the table in section (1) above.

The beginning of the .all.errors file includes explanatory text
describing the format of the lines in the file. Here is an example of
that header for an HBV annotation file:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
  # Each error encountered is printed below, one error per line.
  # Each line has four columns with the following labels:
  #   "accn"         : sequence accession
  #   "idx"          : feature index, full feature names are listed below for each index
  #   "code"         : 3 digit error code
  #   "error-message": error message, possibly with additional information at end enclosed in
"[]"
  #
  # List of features:
  # Feature #1: CDS #1 [single exon; +] polymerase
  # Feature #2: CDS #2 [single exon; +] large envelope protein
  # Feature #3: CDS #3 [single exon; +] middle envelope protein
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
  # Feature #4: CDS #4 [single exon; +] small envelope protein
  # Feature #5: CDS #5 [single exon; +] X protein
  # Feature #6: CDS #6 [single exon; +] pre-capsid protein
  # Feature #7: CDS #7 [single exon; +] capsid protein
  # "N/A" in feature index and desc columns (idx and desc) indicates error pertains to the entire
sequence
  #       as opposed to a specific feature.
  #
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

This section is followed by the list of errors. Here are some example
lines, again from HBV annotation output:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#accn       idx  desc   code  error-message
X59795        1  CDS#1   olp  feature does not overlap with same set of features as in reference
[-(1.1,6.1)]
X59795        6  CDS#6   olp  feature does not overlap with same set of features as in reference
[-(6.1,1.1),-(6.1,7.1)]
X59795        6  CDS#6   trc  in-frame stop codon exists 5' of stop position predicted by
homology to reference [homology search predicted 1795..2433 revised to 1795..1878 (stop shifted
555 nt)]
X59795        7  CDS#7   olp  feature does not overlap with same set of features as in reference
[-(7.1,6.1)]
HQ700584    N/A  N/A     ori  there is not exactly 1 occurrence of origin sequence [0
occurrences]
JQ040171    N/A  N/A     ori  there is not exactly 1 occurrence of origin sequence [0
occurrences]
AF418690      1  CDS#1   olp  feature does not overlap with same set of features as in reference
[-(1.1,5.1)]
AF418690      1  CDS#1   trc  in-frame stop codon exists 5' of stop position predicted by
homology to reference [homology search predicted -697..1623 revised to -697..1055 (stop shifted
568 nt)]
AF418690      5  CDS#5   olp  feature does not overlap with same set of features as in reference
[-(5.1,1.1)]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

The errors are printed in the following order:

For each accession as listed in the input file to the annotation
script:

 - An 'ori' error, if it exists for that accession. (The 'ori' error
   code is unique in that it does not apply to a specific feature
   (CDS/mature peptide) but rather to the sequence as a whole. This
   is why it is listed first.

 - For each feature (CDS or mature peptide) in the order listed in
   the table in (1) above, all error codes that exist for that
   feature in the current accession, if they exist.

If an accession's feature has no error codes associated with it,
there will not be any lines of this file that pertain to that
accession and feature.

If an accession has no error codes associated with it or any of its
features, there will not be any lines of this file that pertain to
that accession.

An overlap (olp) error involves two features. The same overlap error
is printed twice overall, once for each feature, as in the two lines
above:

```
AF418690      1  CDS#1   olp  feature does not overlap with same set of features as in reference
[-(1.1,5.1)]
AF418690      5  CDS#5   olp  feature does not overlap with same set of features as in reference
[-(5.1,1.1)]
```

This follows a general principle that even when the presence of one
error necessarily implies the presence of another error, both errors
are usually printed. A few situations that might be considered
exceptions to this principle are delineated specifically below in
sections (5) through (8).

## (3) Error code information in the .peraccn.errors output file

The annotation script (dnaorg_annotate_genomes.pl) produces an output
file with the suffix <s>.peraccn.errors. This file includes a list of
all accessions with at least one error found, one accession per
line. Each line includes a list of all errors found for that
accession. This file includes less explanatory information on each
error than does the .all.errors file explained in (2) above.

The beginning of the .peraccn.errors file includes explanatory text
describing the format of the lines in the file. Here is an example of
that header for an HBV annotation file:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Each accession for which at least one error was found is printed below.
# One line per accession. Each line is formatted as follows:
#   <accession> <idxA>:<errorcodeA1>(,<errorcodeAM>) <idxN>:<errorcodeN1>(,<errorcodeNM>)
# For indices (<idx>) A to N, each with M error codes.
# Each index refers to a 'feature' in the reference accession as defined below.
# If no index exists, then the error pertains to the entire sequence.
# Feature #1: CDS #1 [single exon; +] polymerase
# Feature #2: CDS #2 [single exon; +] large envelope protein
# Feature #3: CDS #3 [single exon; +] middle envelope protein
# Feature #4: CDS #4 [single exon; +] small envelope protein
# Feature #5: CDS #5 [single exon; +] X protein
# Feature #6: CDS #6 [single exon; +] pre-capsid protein
# Feature #7: CDS #7 [single exon; +] capsid protein
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

This section is followed by the list of accessions with >= 1
error. Here are some example lines, again from HBV annotation output:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
X59795 1:olp:-(1.1,6.1) 6:olp:-(6.1,1.1),-(6.1,7.1),trc 7:olp:-(7.1,6.1)
HQ700584 ori
JQ040171 ori
AF418690 1:olp:-(1.1,5.1),trc 5:olp:-(5.1,1.1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

The accessions will be printed in the order they appear in the input
accession list file passed into the annotation script.

If an accession has no error codes associated with it or any of its
features, there will not be any lines of this file that pertain to
that accession.

The order of the errors per accession is the same as it is for the
.all.errors file described in (2). The order is determined primarily
by feature. First, comes the 'ori' error, if it exists, then all error
codes for feature 1, then all error codes for feature 2, then all
error codes for feature 3, and so on.  Per-feature error codes appear
in the same order they appear in the error code table in (1).  The
error codes for feature i (i is a positive integer), start with 'i:'.
If are no errors for feature i, then the string 'i:' does not appear
for that accession.

As for the .all.errors file, an because an overlap (olp) error
involves two features, the same overlap error is printed twice
overall, once for each feature.

## (4) Thoughts on 'really bad' error combinations that could be used to identify and filter problematic sequences

One overarching objective of the viral annotation project is to
identify objective quality control criteria and to test which viral
genomes meet those quality control criteria. One use of such
evaluations is to identify a subset of genomes that are high quality
enough to merit being in the inputs for downstream analyses such as
classification, in silico genotyping, and building phylogenetic trees.

The three most extreme errors are probably 'nop', 'str' and 'nst'. Any accession with at least one of these errors could be considered 'problematic'. Problematic genomes should probably should be filtered out before analyses such as phylogenetic inference, in which genomes are considered as an ensemble and hence, the presence of a problematic genome can taint the results.

The fourth most extreme is probably 'nm3', although it can occur in valid sequences. One pending suggestion is to identify unique length nm3 errors (that is, features that have a feature with a nm3 error and a length that differs from all other observed lengths for that feature), as a way to label 'problematic' sequences. The reasoning is that recurrent nm3 errors likely indicate natural variation, while unique nm3 errors likely indicate errors in the sequencing or genome assembly.  This remains to be explored.

--------------------------------------------------------------------
## (5) List of required error combinations
--------------------------------------------------------------------
NONE (?)

Previously we thought that:
- ext will only occur in combination with stp
- nst will only occur in combination with stp

However this is not true, because a 'stp' error is only thrown if predicted final 3 nts of a CDS are not a valid stop codon regardless of the frame they are in. 'ext' errors occur if no valid *in-frame* stop codon exists between predicted start and stop, and 'nst' error occurs if no valid *in-frame* stop codon exists 3' of predicted start. So you can have cases of 'ext' and not 'stp', and you can have cases of 'nst' and not 'stp'.

--------------------------------------------------------------------
## (6) List of incompatible combinations of different error codes
--------------------------------------------------------------------
- str will never occur in combination with stp or trc or ext
- trc will never occur in combination with ext or nst or aji or inp
- nop will never occur in combination with nm3 or bd5 or bd3 or str or stp or trc or ext or ntr or nst or aji or int or inp
  (nop will only occur in combination with olp or aja or ajb)

[THIS LIST IS (PROBABLY) NOT EXHAUSTIVE!]

These incompatible combinations are now enforced by the annotation script in the 'dnaorg.pm::initializeHardCodedErrorInfoHash' function. See the code for details.

--------------------------------------------------------------------
## (7) Other known restrictions on combinations of different error codes
--------------------------------------------------------------------
- at most one of trc, ext, nst will be observed for a feature
[THIS LIST IS (PROBABLY) NOT EXHAUSTIVE!]

--------------------------------------------------------------------
## (8) Notes on olp errors:
--------------------------------------------------------------------
The concept of overlap is relevant to any two features that span a non-neglibible genomic interval, such as genes or mature peptides. The concept of overlap is not applied to the origin sequence or the presence/absence of any variants.

- olp errors are further subdivided into
  olp- meaning an expected overlap is absent
  olp+ meaning an overlap was detected when it was not expected

- The str, trc, ext and stp errors do not prevent olp errors from being reported. The predicted start and stop positions of each feature are compared to determine overlaps, even if those features have a str, trc, ext or stp error. However, the first comment in 'Miscellaneous notes' is pertinent to understanding the interaction between trc and olp errors in the same feature.

- nop errors result in no start and stop prediction for a feature, and thereby prevent any test for overlaps for the feature which is not predicted; as a consequence, if feature i has a nop error, then feature i will not overlap with any other feature, and there will be an olp- error if feature i is expected to overlap with another feature. This can cause an olp error, if that feature does overlap with any other features in the reference.

- If two features overlap in the reference, and are instead adjacent in the genome being annotated, then this will lead to olp- errors for both features, an ajb error for the earlier feature, and an aja error for the later feature.

## (9) Notes on stp, trc, ext and nst errors:

- The codes stp, trc, ext, and nst are related.  For exon-based CDS (as opposed to mature peptide-based CDS): at most one of trc, ext, and nst is printed when those three problems arise and the ext and nst errors are always accompanied by a stp error. A trc error can be co-occur with a stp error, but does not have to, which allows us to distinguish between early stops that occur in predictions that have a valid stop in the homologous stop position from those that do not. For mature peptide-based CDS, the relationship between these errors is more complicated because these CDS are annotated based on the collective annotation of the mature peptides that comprise them, and there is not an attempt here to define precisely that relationship.

- The 'trc' and 'ext' error codes differ in the following way:

  o For a 'trc' error the predicted stop will 5' (upstream) of the homology-based stop and the homology-based stop may or not be a valid in-frame stop codon. Thus a 'trc' error may or may not be accompanied by a 'stp' error.

  o For an 'ext' error, the predicted stop will be 3' (downstream) of the homology-based stop and the homology-based stop will *not* be an in-frame stop codon. Thus, an 'ext' error is always accompanied by a 'stp' error.

  o Above, 'homology-based stop' refers to the stop position indicated by the homology search, which is changed in the case of a 'trc' or 'ext' error as described in 'Miscellaneous notes' below

## (10) Miscellaneous notes

- In the current implementation a stop position is predicted first, by analysis of homology-search based nucleotide alignments and models, and refined second, by analysis of translation of the nucleotides to amino acids. Therefore, the predicted stop positions are not necessarily the positions determined by the homology-based nucleotide analysis alone. Often they are, but in the case of a 'trc' or 'ext' error, the homology-based stop position will have been 'corrected' to be the first in-frame stop codon 3' (downstream) of the predicted start codon (which has been validated to be an ATG).