

Splitting and Oring with L2Groups
Jim Linnemann

1. Valid triggers in the L2Groups database are 5-tuples of the form:

IndexNumber, TriggerName, L1Script, L2Group, L3Script

2. Valid triggers in the TDB with *Or markers can be mechanically converted to this form. These input 5-plets are of the form

IndexNumber, TriggerName, L1Script, L2, L3

Where

L2 = {L2Script | L2_OR_MARKER}

L3 = {L3Script | L3_OR_MARKER}

3. All names are actually name/version pairs.

4. Assumptions: This conversion assumes that conversion will be attempted ONLY for triggers which are pure Splitting, or pure Oring. Also, it is assumed that no L1 separators have been used. These two complications both can be prevented by choice of the IndexNumber.

5. Splitting Conversion.

For pure splitting, the only change which needs to be made is that L2Scripts must be converted to L2Groups. This is done as follows:

5.1 An input 5-plet is re-emitted as a 5-plet with the L2Script entry changed to a L2Group entry with the same name and version.

5.2 If no existing L2Group entry of that name/version exists, it is created, with a name equal to the name/version of the L2Script, and it contains a single L2Script entry equal to the L2Script name/version. The Description of the new L2Group is a copy of the Description of the L2Script.

5.4 The TriggerName and its Description are unchanged.

5.5 The xml generated for Splitting Trigger from a database in the L2Group schema is to be identical in all respects to the xml generated from the corresponding Trigger in the original schema. Pre-version 15 trigger lists with no L2 branching are specific examples: all of their triggers are pure Splitting Triggers.

6. Oring Conversion.

6.1 Input form: L2Or group and L3Or group

For pure oring, the sequence of 5-plets consists of 2 blocks. The first is an L2Or group consisting of lines of the form:

IndexNumber, TriggerName, L1Script, L2Script, L3_OR_MARKER

in which the L1Script is the same in each line

The second is an L3Or group consisting of lines of the form:

IndexNumber, TriggerName, L1Script, L2_OR_MARKER, L3Script

in which the L1Script is the same in each line, and the same as the L1Script in the L2Or group block..

Example:

| | | | | |
|----|----|------|--------------|--------------|
| 7 | L1 | TN_B | L2B | L3_OR_MARKER |
| 8 | L1 | TN_C | L2C | L3_OR_MARKER |
| 9 | L1 | TN_D | L2D | L3_OR_MARKER |
| 10 | L1 | TN_a | L2_OR_MARKER | L3a |
| 11 | L1 | TN_b | L2_OR_MARKER | L3b |

This Oring trigger contains a 3-script L2Or group with scripts L2B, L2C, L2D, and 2-script L3Or group with members L3a, L3b .

6.2 Output form of L2Or group

The L2Or group is not re-emitted as trigger 5-tuples. Instead, it is recognized and mechanically converted into a reference to a L2Group to be used by the L3Or group to follow. If the L2Group does not exist, it is created.

6.2.1 L2Group name/version choice

The L2Group name is formed by a “L2OR(” prefix followed by the included L2 script names separated by commas, and terminated by a “)””.

The database is searched for a L2Group of that name. If the version numbers of the contents do not match, a new version of the L2Group will be created; otherwise the current version is used. If this L2Group does not yet exist, then version 1 is created.

In the example above, the L2Group name would be L2OR(L2B,L2C,L2D). If no L2Group with content [L2B, L2C, L2D] already existed, it would be created with version 1.

This algorithm assumes that L2Groups with more than one member will be formed according to this convention, removing the necessity of searching contents of all L2Groups to prevent duplication. The usual constraints on names/versions in a trigger list will apply to L2Groups.

6.2.3 L2Group Descriptions

The Description of an L2Group will be formed from a heading and the Descriptions of the L2Scripts it contains with numbered separators between the Descriptions. In our example with three L2Scripts, this would look like the following, with the generated text in **bold** and the copied L2ScriptDescription text in *italic*.

OR of the following conditions:

- 1) *the Description of the L2Script L2B*
- 2) *the Description of the L2Script L2C*
- 3) *the Description of the L2Script L2D*

6.3 Output form of L3Group

As mentioned above, the only Trigger output would be 5-plets corresponding directly to the L3Or group, with the relevant L2Group name substituted for L2_OR_MARKER. The emitted 5-plets would be:

| | | | | |
|---|----|------|--------------------------|-----|
| 7 | L1 | TN_a | L2OR(L2B,L2C,L2D) | L3a |
| 8 | L1 | TN_b | L2OR(L2B,L2C,L2D) | L3b |

Notice that this reduces the number of 5-tuples from the input of (L2Orgroupsize + L3Orgroupsize) to L3Orgroupsize lines, here from 3+2 to 2 lines. The TriggerNames of the L3Or group are used unaltered. Then the generated xml for the original and transformed list should be identical.

6.3.1 Trigger Descriptions of the L3Group, however, are modified by substituting the Description of the L2Group for the uninteresting L2 Description text of the L2_OR_MARKER. This requires parsing and manipulating the Description associated with the TriggerName, which is described in the next section.

6.3.2 Form of TriggerName Descriptions.

These Descriptions are hand-edited, but follow a strict format, with the relevant tags denoted as **bold** below and the relevant text in *italic*:

L1: *L1_descriptive_text*

L2: *L2_descriptive_text*

L3: *L3_descriptive_text*

The *descriptive text* is usually closely related or identical to the corresponding Description of the relevant Script member of the Trigger. To replace the L2 descriptive text in the TriggerName Description, the code will need to extract the TriggerName Description, then find the L2 part of the description by locating the beginning starting with “L2:”, and the end, marked by “L3:”, and replace the removed *L2_descriptive_text* with the text of the Description of the L2Group now a member of this TriggerName (instead of the fake L2_OR_MARKER L2Script).

7. Backwards Compatibility and Operation Modes

Because the Trigger database is historical, it is important that one be able to accurately reproduce xml for previously-used trigger lists which have been used in data taking. In particular, the order of bits must be preserved (or reconstructed). Two common issues are 1) understanding the scripts for a given bit and 2) programming a trigger simulator to determine trigger behavior for data taken with a specific trigger list. D0 has taken data with versions 15.0 through 15.04 trigger lists which use the Or marker notation which must be supported in this way even after the TDB has evolved to the L2Group schema. To achieve this, we need the following functionality in addition to that outlined above.

7.1 Intelligent Trigger List Porting

This mode will perform on specific trigger lists the full schema porting described in sections 1-6.

In this mode, it is understood that the `tl_index` numbers would change, and thus names would change as well, even though the ported list is logically identical to the original one. Further, since L2Groups are unordered lists, the L2Group member scripts/bits will not be in the same order as the corresponding L2 scripts in the Or marker version. Rather, they are likely to be in alphabetical order in both the L2Group name and the xml. If it were felt to be really important to retain order, the porting tool might pre-pend an ordering number to the script name, but this could interfere with maintaining uniqueness (having a single L2Group for each collection of Or'd L2 scripts, no matter how many times the group is used).

7.2 Literal Porting

The intention of this (new) functionality is to produce a literal transcription of historical trigger lists which could exactly re-produce the original xml.

In this mode, instead of producing L2Groups from a group of Or marker scripts as described in sections 1-6, instead, each individual L2 script with a L3 Or marker will be processed as if it were a Splitting script, producing (or re-using as appropriate) a single-script L2Group as in 5.1 above. Similarly, the L2 Or Markers with real L3 scripts will be re-emitted with a reference to a L2Group consisting solely of the L2 Or Marker. That is, the output is the trigger list still in Or Marker notation, but implemented in the L2Group database schema.

Xmlgen thus must retain the ability to generate xml from the resulting Or marker scripts, and the xml generated should be identical in all respects to that generated by the original trigger list before the L2Group schema change. This will meet the requirement of being able to reproduce the original online bit ordering and xml.

7.3 Operational Considerations

After porting, *any* trigger list, V1 through V15, the physics results—whether any individual event passes or fails—should still be identical at all triggering levels.

Note that either Intelligent or Literal porting (7.1 or 7.2) of a trigger list should produce identical results (both in xml, and in bit ordering) for trigger list version 1.0-14.94. The only differences between Literal and Intelligent porting for xml, names, or bit ordering are restricted to V15 trigger lists produced since the introduction of *Or markers.

It should be possible to convert the entire database with the porting tool (or a script running it) producing a new database with all trigger lists in the new L2Groups schema. Perhaps full conversion under literal mode first is most natural. Then Intelligent mode could be applied selectively to specific V15 scripts to be added to the L2Group-schema database, with new version numbers.

If this scheme is chosen, the command line arguments for Intelligent mode could specify the trigger list and input version to be ported, and the trigger list and version to produce as an output. The output version should be different than the input version to avoid attempting to overwrite the Literally-ported version.

7.3 Likely Use Case

Suppose that these facilities are used when the V15.04 trigger list is current.

A likely use of these facilities would use Literal porting (section 7.2) to produce from (for example) all trigger lists from V1.0 through V14.94 (these were all before L2 branching and thus before Or Markers); and then V15.0 through V15.04 trigger lists, all ported into the L2Group schema database. Each ported trigger list would be capable of producing identical xml as they would have in the previous database schema. At this point we would own the original database in the original schema and a “copy in the new schema, each containing all trigger lists.

Then V15.0 through V15.04 trigger lists would be ported from the original database with Intelligent porting (section 7.1) and added to the new database, named as versions V15.10 through V15.14, so they could also be viewed by users and Meisters in the more elegant and logical L2Groups notation. Finally we’d produce from V15.04 aV15.20 where the V15.20 would be the starting point for further development of the trigger list in L2Group format. When all this was demonstrated to be working, future work would all take place in the database with the new L2Group schema.