How to modify the HLT TCK

Introduction to TCKs and Manifests

The Hlt configuration lives in $HLTTCKROOT/config.tar, which contains all configurations ever deployed. In addition to providing the configurations, this file is also used as the source for the so-called manifests. These manifests enumerate, for the various versions of Moore, which configurations are available, and are generated from the config.tar file by executing (after setting up the right environment) 'cmt make'. This generates the set of files The PVSS panel in the pit which allows one to select a TCK uses these manifests to present the list of available configurations for a given release of Moore.

So in order to be able to select a configuration in PVSS, first it needs to exists (i.e. be present in $HLTTCKROOT/config.tar) and second it needs to be in the relevant manifest file. In the online system, Moore is installed under /group/hlt/MOORE/Moore_vxrypz, and the manifest for that version of Moore will be present in /group/hlt/MOORE/Moore_vxrypz/InstallArea/TCK/manifest. But note that $HLTTCKROOT/config.tar contains all configurations for all versions -- and there is one manifest per version (patch versions are lumped together with the vxry versions). So one sees for example:

```
[graven@plus06 ~]$ cd /group/hlt/MOORE/Moore_v9r1dev
[graven@plus06 Moore_v9r1dev]$ ls -l TCK/HltTCK/config.tar
-rw-r--r-- 1 hlt_oper hlt 63305728 Apr 29 15:27 TCK/HltTCK/config.tar
```

and the manifest files which are generated from it:

```
[graven@plus06 Moore_v9r1dev]$ ls  TCK/HltTCK/manifest
MOORE_v5r1  MOORE_v5r3  MOORE_v5r5  MOORE_v6r0  MOORE_v6r2  MOORE_v7r1  MOORE_v7r3  MOORE_v7r5  M
MOORE_v5r2  MOORE_v5r4  MOORE_v5r6  MOORE_v6r1  MOORE_v7r0  MOORE_v7r2  MOORE_v7r4  MOORE_v8r0  M
```

with for example for v9r1:

```
[graven@plus06 Moore_v9r1dev]$ cat TCK/HltTCK/manifest/MOORE_v9r1
Commissioning_OTCosmics : 0x80850000 : bb8e87d744b625879c8b13b746a5b006 : Default
         PassThrough : 0x80840000 : 8d105fa9447104c2ca832d6ea13f0f3e : ODINPhys,Timing acc=1, ODI
         PassThrough : 0x80830000 : d684c7e76cbf23c1ff823afa5fa36973 : ODINPhys,Timing acc=1, ODI
Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10 : 0x00071810 : 6213e73147d43610fb79c1c49601ae6c : Default
Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10 : 0x00071710 : 70139c7b480d34ecaa5a2b9ed9014a43 : Disable L0
Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10 : 0x00071610 : 122d3b692206d4ae56d21661898ef6a0 : Disable L0
Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10 : 0x00071510 : 576c481d7ce326dd152fcef8ab9bf684 : Disable L0
Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10 : 0x00071410 : 47cbb6b186d5ae391d22f6636510bd9e : Disable L0
Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10 : 0x00071310 : 0e119b6346c1dc70e7f50286c25c9409 : Disable L0
Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10 : 0x00071210 : 5961c3adfe61bd54d151163d4ae7859d : Disable L0
Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10 : 0x00051810 : aea292b6e4b319f2e2c3488163522a6
Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10 : 0x00051710 : 35cb212283d63eb323d40d39d74c315
Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10 : 0x00051610 : 1f4654245dc966db9af5ffbc36a4e52
Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10 : 0x00051510 : dbf70bddda2d4160c579a574eb106d0
Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10 : 0x00051410 : e8bbaff1d329f06250237e452f27d32
Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10 : 0x00051310 : 52e2b7c0772b90f96edc4400dc4eb5e
Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10 : 0x00051210 : 39044f6c6a79df8877b8661696708eb
```

Now, the TCK/HltTCK package is agnostic of which version of Moore is actually used, so it just generates all manifests that it can. On the other hand, Moore itself knows it own version number, and when the Hlt/Moore package is build, it will copy the relevant manifest to the InstallArea for PVSS to pick it up. Hence one sees:

```
[graven@plus06 Moore_v9r1dev]$ ls -l  InstallArea/TCK/manifest
-rw-rw-r-- 1 hlt_oper hlt 1948 May  4 09:21 InstallArea/TCK/manifest
[graven@plus06 Moore_v9r1dev]$ diff InstallArea/TCK/manifest TCK/HltTCK/manifest/MOORE_v9r1
[graven@plus06 Moore_v9r1dev]$
```

So to make PVSS aware of any new configuration, one needs 1) generate all manifests 2) make sure the right manifest is copied into the InstallArea

But before one can do anything, one needs to set up the right environment:

```
su - hlt_oper
cd /group/hlt/MOORE
source setup.sh v9r1dev
```

Note: please be extremely careful with the above, and explicitly verify that you set up your environment for the version of Moore you intended. Making a mistake at this point can destroy an installed version of Moore, and hence make datataking IMPOSSIBLE, and it will take (potentially a lot of) time to re-install / fix any screw ups made at this point by eg. picking the 'live' version when you think you're using a 'dev' version. So please be extremely paranoid, and check anything you type. In case you're not sure, do NOT continue.

Now we can rebuild all manifests (local to HltTCK):

```
cd $HLTTCKROOT/cmt
cmt make
```

which covers step 1) in the above, and then we need to rebuild Moore in order to copy the right manifest to the InstallArea:

```
cd $MOOREROOT/cmt
cmt make
```

That covers step 2, and at this point all configurations have been made available to PVSS. However, provided that the manifests were up to date, the above will not have done anything new, as no new configurations have been made.

Use Case : Installing a new released HLTTCK configuration in the pit

A common use case for the HLT piquet is having to install a new HLTTCK configuration for an existing MOORE configuration, as a prelude to changing the recipes in the PVSS panel. In this case

- Log into plus as hlt_oper
- Set up the correct environment

```
cd /group/hlt/MOORE
source setup.sh v12r4
cd $HLTTCKROOT
```

- **Paranoia step**: check where you are! You should be in /group/hlt/MOORE/Moore_vXrY/TCK/HltTCK, where "X" and "Y" correspond to the Moore version you setup earlier. Also, use svn info to check which version of HLTTCK you are currently using, and that everything is as you expect it to be.

```
pwd
svn info
```

- Update to the version of HLTTCK you want to use

```
svn switch http://svnweb.cern.ch/guest/lhcb/DBASE/tags/TCK/HltTCK/v2r2p1
```

- Make the new manifest

```
cd cmt
```

```
cmt make
```

- **Paranoia step**: check what you just did! Do a diff with the install area to see what new things you have made

```
cd ../manifest
diff MOORE_v12r4 ../../../InstallArea/TCK/manifest
```

- Finally, you now have a choice between the paranoid and the tidy way to complete your task. You may either (paranoid) copy your new manifest directly to the InstallArea, in which case you will do the minimum needed to make your new configurations available, but will not verify that someone has not modified another part of the system by hand without checking their changes into SVN. If you want to be tidy, you can use CMT to propagate your new manifest, but in this case CMT will also make sure that the rest of the system is as has been released into SVN, overwriting any local "by hand" modifications.
- **Paranoid solution**

```
cp MOORE_v12r4   ../../../InstallArea/TCK/manifest
```

- **Tidy solution**

```
cd $MOOREROOT/cmt
cmt make
```

Use Case : Creating a new configuration

So how does one create a new configuration? There are two qualitative ways of doing this:

a) create one 'from scratch' b) create a modified version of an already existing configuration

we will not cover a) in this note, except to mention that this is done by running Moore with a special option which dumps the configuration into the config.tar file. Here we will only consider creating a slightly modified version of existing configuration.

So the first step is to decide *what* you want to modify. Let's see that we want to change the prescale factor of some Hlt1 line. Let's start by browsing the set of available configurations interactively by starting 'TCKsh':

```
TCKsh
```

at this point you have a python prompt, and some utilities have been preloaded for you. To get a list of all configurations, one can type

```
listConfigurations()
```

and get (not all output shown):

```
MOORE_v9r1
    Commissioning_OTCosmics
      0x80850000 : bb8e87d744b625879c8b13b746a5b006 : Default
    PassThrough
      0x80830000 : d684c7e76cbf23c1ff823afa5fa36973 : ODINPhys,Timing acc=1, ODINTech,Aux,NZS,Cal
      0x80840000 : 8d105fa9447104c2ca832d6ea13f0f3e : ODINPhys,Timing acc=1, ODINTech,Aux,NZS,Cal
    Physics_25Vis_25L0_2Hlt1_2Hlt2_Apr10
      0x00071210 : 5961c3adfe61bd54d151163d4ae7859d : Disable L0-SPD,PU,FullBias
      0x00071710 : 70139c7b480d34ecaa5a2b9ed9014a43 : Disable L0-SPD
      0x00071310 : 0e119b6346c1dc70e7f50286c25c9409 : Disable L0-SPD,PU20,FullBias
      0x00061810 : 11d0c37fcedd7233b81854559044af30 : Default
      0x00061610 : 6ef63c74fd3ac14eb0af1d482d7e05c5 : Disable L0-PU
      0x00071610 : 122d3b692206d4ae56d21661898ef6a0 : Disable L0-PU
      0x00071510 : 576c481d7ce326dd152fcef8ab9bf684 : Disable L0-SPD,PU20
      0x00061510 : 2ae17f26aa8df4d64a458338edb8f7ee : Disable L0-SPD,PU20
      0x00071810 : 6213e73147d43610fb79c1c49601ae6c : Default
      0x00071410 : 47cbb6b186d5ae391d22f6636510bd9e : Disable L0-SPD,PU
      0x00061410 : 640df02aa541caa69498abe49da6b612 : Disable L0-SPD,PU
      0x00061710 : 13bcdbdf7615347b4dd36fae5bbe036c : Disable L0-SPD
      0x00061310 : 219aaec32bd0eb4e59d1d19be77cc5f5 : Disable L0-SPD,PU20,FullBias
      0x00061210 : 6cc95035b58e12c0c5f11bb58f404911 : Disable L0-SPD,PU,FullBias
    Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10
      0x00051410 : e8bbaff1d329f06250237e452f27d326 : Disable L0-SPD,PU
      0x00051710 : 35cb212283d63eb323d40d39d74c3153 : Disable L0-SPD
      0x00051510 : dbf70bddda2d4160c579a574eb106d0e : Disable L0-SPD,PU20
      0x00051310 : 52e2b7c0772b90f96edc4400dc4eb5e1 : Disable L0-SPD,PU20,FullBias
      0x00051610 : 1f4654245dc966db9af5ffbc36a4e52c : Disable L0-PU
      0x00051210 : 39044f6c6a79df8877b8661696708eb5 : Disable L0-SPD,PU,FullBias
      0x00051810 : aea292b6e4b319f2e2c3488163522a65 : Default
```

wait, there are more configurations here than appeared in the manifest! How can this be? Well, the script which generates the manifest can 'veto' certain configurations, but those configurations have been archived. The reason for this is that we do not want the 0x0006abcd (with abcd in [0x1210, 0x1810]) to be available for future running, as it has been superseeded by 0x0007abcd. But since 0x0006abcd was released at some point (and maybe data was taken with this configuration), it can not be deleted, just be made unavailable to PVSS.

The second observation is that in addition to the TCK there is a 128 bit number in the form of a 32 character hex string. This number is computed from the contents of the configuration, and this is how internally configurations are addressed -- if one ever makes a configuration which already existed identically, you get the same value. So Moore picks up the TCK, then maps this onto this large number, and from then on only uses this 128 bit number for its internal bookkeeping.

The configurations contain every single property of every single algorithm and tool used by the Hlt sequencer. So one can ask for example for all the prescale factors, given a certain TCK:

```
>>> listProperties(0x00051810,'Hlt1L0.*','AcceptFraction')

   Requested Properties for DeterministicPrescaler/Hlt1L0MuonPreScaler
      'AcceptFraction':1

   Requested Properties for DeterministicPrescaler/Hlt1L0MuonPostScaler
      'AcceptFraction':1e-06

   Requested Properties for DeterministicPrescaler/Hlt1L0MUON,minbiasPreScaler
      'AcceptFraction':1

   Requested Properties for DeterministicPrescaler/Hlt1L0MUON,minbiasPostScaler
      'AcceptFraction':1e-06
```

(not all output shown). Note that ListProperties takes three arguments:

1) the configuration 2) the component name (a component is an algorithm, tool, service) 3) the property name

and that one can provide regular expressions for the 2nd and 3rd argument.

Let's decide that we want to change the 'AcceptFraction' property of the Hlt1L0MuonPostScaler which is now 1e-06, and we want to change this to 1e-02. We start by looking at the 128 bit ID corresponding to 0x00051810, which listConfigurations() told us is 'aea292b6e4b319f2e2c3488163522a65'. We can now create an updated version of this configuration:

```
updateProperties( id = 'aea292b6e4b319f2e2c3488163522a65'
                                , updates = { 'Hlt1L0MuonPostScaler' : { 'AcceptFraction' : '1e-0
                                , label = 'Default, L0MuonPostScale = 0.01'
```

```
                                        , cas = ConfigTarFileAccessSvc(Mode='ReadWrite')
                                        )
```

Going through the arguments, the first, id, specifies which configuration to start from. The second, updates, specifies what to change, the third, label, specifies the label that the shifter will see in the PVSS selection box, and the last option specifies that we want to write to the tar file (the default is read-only!).

So the most interesting bit is the 'updates' part. This argument is a dictionary of dictionaries. The idea is as follows: one may want to update several components (algorithms, tools) at the same time, so you need to specify which components. This is done by providing a dictionary which maps the name of a component to the changes you want to make for that component. Hence, updates is a dictionary, with the 'key' corresponding to the component to be changed, and the corresponding 'value' to the changes you want to make for that component.

But the changes you want to make are again a dictionary, with as 'key' the name of the property to be changes, and the 'value' the requested value of property specified in the 'key'. So one can change multiple properties, of multiple components all at the same time.

At this point, a word of caution is needed: in general, a fast run change will be allowed between configurations which are thus created. This implies that from one run to the next, the properties will be updated -- but not all components actually implement this properly. So it is possible to make an updated configuration, that, if one starts with it, does exactly what you want, but if you start with the original, and then make a fast run change to the new one, the algorithm (tool) will just ignore that the property has been updated 'behind its back'. Hence one needs to be aware that the code is compatible with the change in configuration. Fortunately, the DeterministicPrescaler supports changes to its AcceptFraction property during fast run changes 😉

Returning to the 'updatedateProperties' command above -- if you run it from the TCKsh prompt, you get:

```
updating: Hlt1L0MuonPostScaler.AcceptFraction:1e-02
 configTree(e7b7af8df9a962879e6650c75fcfeb82)::addParent(937f9209084efdea6656a206bf537b14) alread
wrote TOPLEVEL/MOORE_v9r1/Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10/27a5ebdfd2dd2188cf8d
'27a5ebdfd2dd2188cf8d76fcb5355bd4'
```

where the 'id' of the newly made configuration is shown. You could also assign this for later use to a python variable:

```
>>> id = updateProperties( id = 'aea292b6e4b319f2e2c3488163522a65'
...                                 , updates = { 'Hlt1L0MuonPostScaler' : { 'AcceptFraction' : '
...                                 , label = 'Default, L0MuonPostScale = 0.01'
...                                 , cas = ConfigTarFileAccessSvc(Mode='ReadWrite')
...                                 )
updating: Hlt1L0MuonPostScaler.AcceptFraction:1e-02

 configTree(e7b7af8df9a962879e6650c75fcfeb82)::addParent(937f9209084efdea6656a206bf537b14) alread
wrote TOPLEVEL/MOORE_v9r1/Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10/27a5ebdfd2dd2188cf8d
>>>
>>> id
'27a5ebdfd2dd2188cf8d76fcb5355bd4'
```

Let's first check that we really got what we wanted by comparing the two configurations:

```
>>> diff( 'aea292b6e4b319f2e2c3488163522a65','27a5ebdfd2dd2188cf8d76fcb5355bd4')
--- DeterministicPrescaler/Hlt1L0MuonPostScaler (IAlgorithm) aea292b6e4b319f2e2c3488163522a65
+++ DeterministicPrescaler/Hlt1L0MuonPostScaler (IAlgorithm) 27a5ebdfd2dd2188cf8d76fcb5355bd4
@@ -27,1 +27,1 @@
-'AcceptFraction':1e-06
+'AcceptFraction':1e-02
```

```
>>>
```

this shows the difference between the original and the new configuration, and indeed the only difference is in the AcceptFraction of the Hlt1L0MuonPostScaler algorithm.

Now we can assign a TCK to this configuration. This is done using the 'createTCKEntries' command from TCKsh. It requires a dictionary where the keys correspond to the TCK, and the corresponding value to the ID, and it again requires write access to the config.tar file. Let's start by doing something you shouldn't be able to do:

```
>>> createTCKEntries({ 0x00051210:id } , cas = ConfigAccessSvc(Mode='ReadWrite') )
KeyError('requested L0TCK 0x1210 not known by L0DUConfigProvider in config 27a5ebdfd2dd2188cf8d76
>>>
```

What we tried was to force this configuration to use an L0 TCK of 0x1210 -- but this particular configuration only knows about 0x1810 (because the original we made it from only knows about 0x1810) and hence this is refused. So we can only assign a TCK ending in 0x1810 to this configuration. Lets try again:

```
>>> createTCKEntries({ 0x00051810:id } , cas = ConfigAccessSvc(Mode='ReadWrite') )
creating mapping TCK: 0x00051810 -> ID: 27a5ebdfd2dd2188cf8d76fcb5355bd4
ConfigTarFileAc...  ERROR  Alias already exists, but contents differ... refusing to change
```

This time, we asked it to assign 0x00051810 to our new configuration -- but that TCK already exists, and points to a different configuration. Hence this is also refused. So we now pick a new TCK for this:

```
>>> createTCKEntries({ 0x00091810:id } , cas = ConfigAccessSvc(Mode='ReadWrite') )
creating mapping TCK: 0x00091810 -> ID: 27a5ebdfd2dd2188cf8d76fcb5355bd4
>>>
```

and now lets check using 'listConfigurations()' that this TCK exists (note: not all output shown):

```
    Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10
      0x00091810 : 27a5ebdfd2dd2188cf8d76fcb5355bd4 : Default, L0MuonPostScale = 0.01
      0x00051410 : e8bbaff1d329f06250237e452f27d326 : Disable L0-SPD,PU
      0x00051710 : 35cb212283d63eb323d40d39d74c3153 : Disable L0-SPD
      0x00051510 : dbf70bddda2d4160c579a574eb106d0e : Disable L0-SPD,PU20
      0x00051310 : 52e2b7c0772b90f96edc4400dc4eb5e1 : Disable L0-SPD,PU20,FullBias
      0x00051610 : 1f4654245dc966db9af5ffbc36a4e52c : Disable L0-PU
      0x00051210 : 39044f6c6a79df8877b8661696708eb5 : Disable L0-SPD,PU,FullBias
      0x00051810 : aea292b6e4b319f2e2c3488163522a65 : Default
```

Note that at after doing this, we need to inform PVSS about this, by running the steps 1) and 2) outlined at the top.

Now lets do something slightly more complicated (but basically the same) and change the L0TCK of this configuration. This one could do by hand, using updateProperties, and making sure you get all the properties of the L0DUConfigProvider correct. That would be error prone, so instead there is a utility that does this for you. It will read all the L0 configurations from $L0TCK/... and pick up all the right properties:

```
>>> id = updateL0TCK( '27a5ebdfd2dd2188cf8d76fcb5355bd4', '0x1210', 'Disable L0-SPD,PU,FullBias,
key: Channels  request: [['name=[HCAL]', 'rate==[100.0]', 'conditions= [Hadron(Et)>12]', 'MASK=[0
key: TCK  request: 0x1210  persistent: 0x1810
key: Conditions  request: [['name=[Electron(Et)>35]', 'data=[Electron(Et)]', 'comparator=[>]', 't
key: Description  request: TCK = 0x1210 : L0-Mb v1 (PU OFF) + L0-Fb (OFF) + L0-Bg  persistent: TC
key: Name  request: MINBIAS_Feb2010_TCK1210_0x1210  persistent: MINBIAS_Feb2010_TCK1810_0x1810
requested update: <ROOT.vector<string> object at 0x95e7e60>
 configTree(e7b7af8df9a962879e6650c75fcfeb82)::addParent(937f9209084efdea6656a206bf537b14) alread
wrote TOPLEVEL/MOORE_v9r1/Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10/0ba8631555d53d728a44
>>>
```

as you can see, you call 'updateL0TCK' with the id of the configuration you want to start from, the L0 TCK you want the new configuration to use, the label for the new configuration, and again you need to tell it you want write access to the config.tar file.

The code generated automatically the relevant updates, and generated a new configuration:

```
>>> id
'0ba8631555d53d728a44bf6fd9968560'
```

and we can again assign a TCK to this id:

```
>>> createTCKEntries({0x00091210:'0ba8631555d53d728a44bf6fd9968560'},cas = ConfigAccessSvc(Mode='
creating mapping TCK: 0x00091210 -> ID: 0ba8631555d53d728a44bf6fd9968560
```

And again, listConfigurations() shows the result:

```
    Physics_MinBiasL0_PassThroughHlt_ExpressHlt2_Apr10
      0x00091810 : 27a5ebdfd2dd2188cf8d76fcb5355bd4 : Default, L0MuonPostScale = 0.01
      0x00091210 : 0ba8631555d53d728a44bf6fd9968560 : Disable L0-SPD,PU,FullBias, L0MuonPostScale
```

And do not forget to update the manifests as outlined at the top of this note!

Note that one can use any L0 configuration which is known in $L0TCK, so if a new L0 configuration is made, one should 'getpack' the new version of TCK/L0TCK into the Moore release, i.e. in this case

```
su - hlt_oper
cd /group/hlt/MOORE/Moore_v9r1dev
getpack --user=yourUserNameGoesHere TCK/L0TCK vxry

cd /group/hlt/MOORE
source setup.sh v9r1dev
```

and then run TCKsh in order to see this new version of L0TCK. (or, if the new L0TCK is already in /sw/lib, the last two commands should pick it up).

Note that the 'updateL0TCK' only updates the L0DUConfigProvider, and that the L0DUConfigProvider has been instrumented at run changes to pick up the updated properties -- but that does not imply that one has a consistent configuration which makes sense. For example, one could have updated the L0TCK to 0xDEB2 and technically that would work, but the resulting configuration would be nonsense. So please be careful that any configuration generated actually makes sense.

</verbatim>

-- EricvanHerwijnen - 11-May-2010

How do I run the HLT2 without PID selections?

-- MichaelWilkinson - 2018-08-22

This topic: LHCb > HLTTCK
Topic revision: r7 - 2018-08-22 - MichaelKentWilkinson