

ATLAS Software tutorial  
21<sup>st</sup> – 23<sup>rd</sup> October 2009

# Trigger Data for Analysis

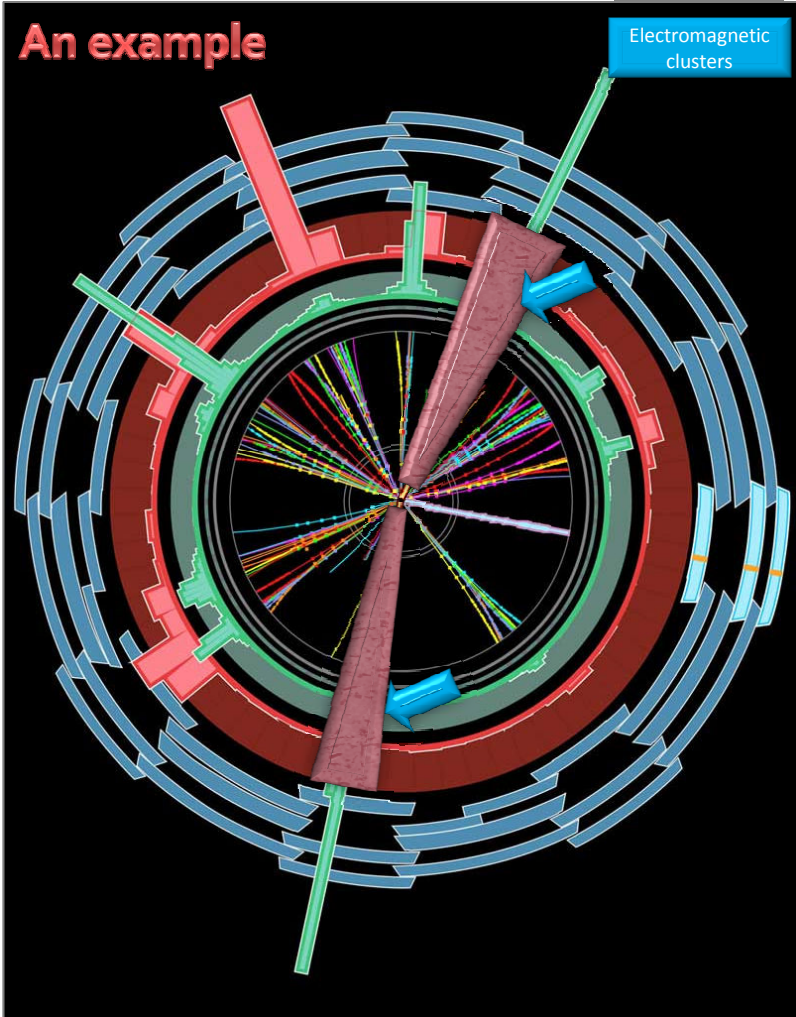
*Jörg Stelzer DESY*

*Ricardo Gonalo RHUL*

# The Trigger

EM ROI

An example



Level1:  
Region of Interest is found and position in EM calorimeter is passed to Level 2

Level 2 seeded by Level 1

- Fast reconstruction algorithms
- Reconstruction within ROI

Ev.Filter seeded by Level 2

- Offline reconstruction algorithms
- Refined alignment and calibration

L2 calorim.

cluster?

L2 tracking

track?

match?

E.F.calorim.

E.F.tracking

track?

e/ $\gamma$  reconstr.

e/ $\gamma$  OK?

# The Trigger Execution

**FEX** FEX algo's are executed to create features on which selection in HYPO algo's is based

## Chain:

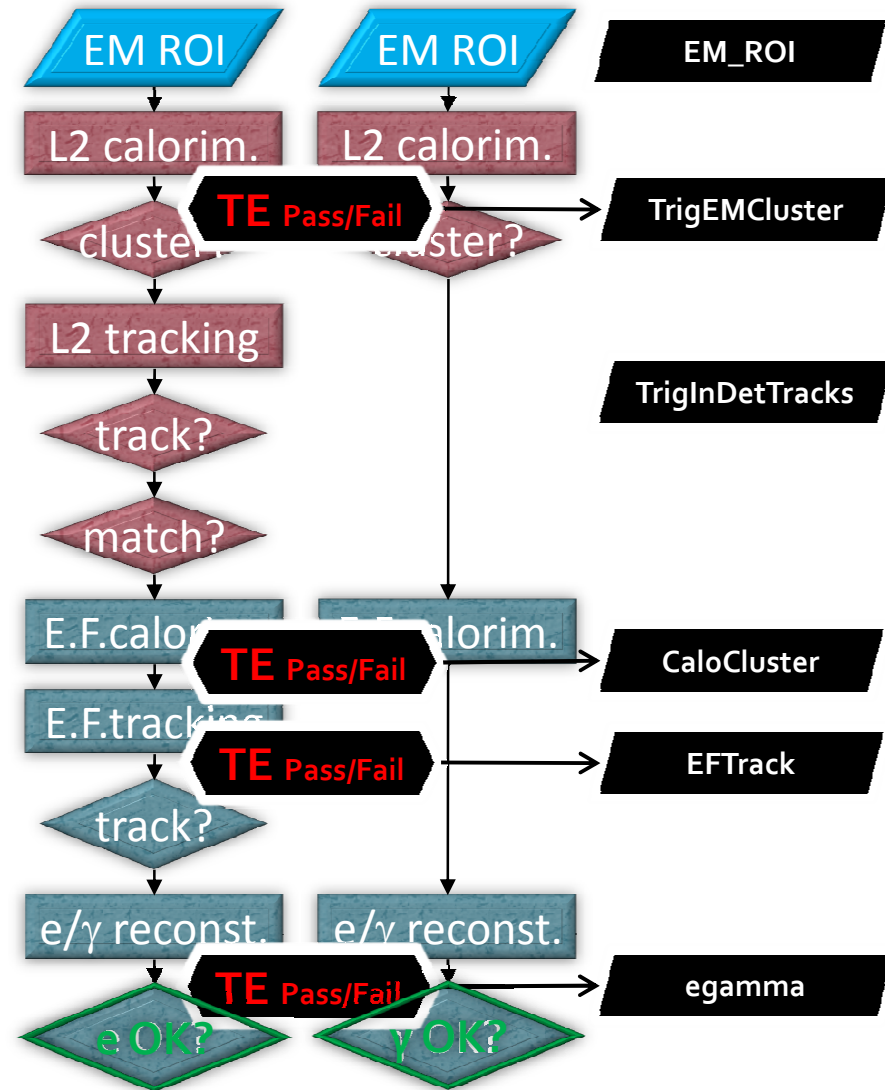
- Started, if seed has fired and chain is not **PRESCALED**
- Stopped **AT STEP**, if a HYPO is not passed
- Last HYPO passed → **CHAIN PASSED**

## Event:

- Passed, if at least one EF chain is passed
- Put into all streams that are associated with the passed EF chains

- Trigger objects associated with triggers through Navigation **TriggerElements**

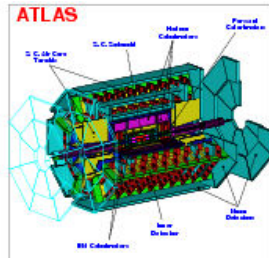
- Trigger information in **ESD** **AOD** **DPD** **TAG**
  - TRIGGER DECISION** + Configuration + Navigation + Feature



# Trigger Configuration + Data

Preparation

Data taking



Configures

TriggerDB  
All configuration data

Stores decoded  
Trigger Menu

Online  
Conditions  
Database  
COOL

Encoded trigger result  
from all 3 levels

Decoded trigger menu

Reconstruction

Trigger aware  
analysis

Trigger Result

- passed?, passed through?, prescaled?, last successful step in trigger execution?

Trigger EDM

- Trigger objects for trigger selection studies

Trigger Configuration

- Trigger names, prescales, pass throughs

access through TrigDecisionTool

ESD

AOD

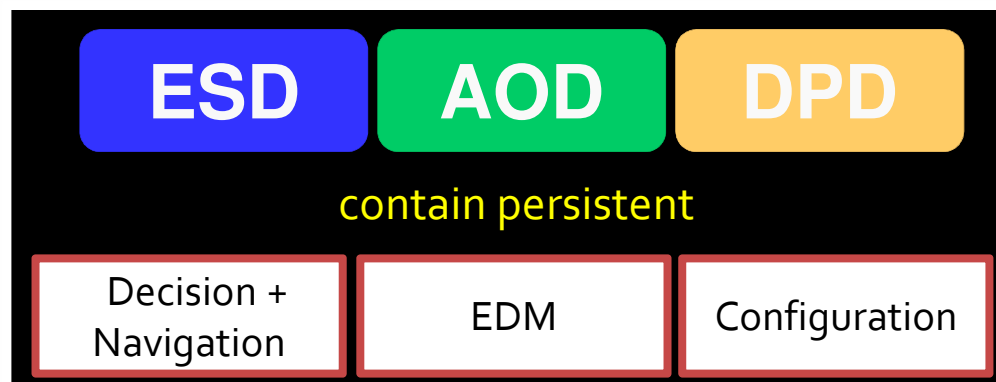
DPD

TAG

With decreasing  
amount of detail

- Analysis based on single trigger chain or an 'OR' of a few chains (a 'ChainGroup')
- Chain definition – algorithms, cuts, multiplicities – do not change during a run, but can change between runs
  - Important for analysis on DPD, where multiple runs are merged, check the ChainGroup content between each run
- Prescales at LVL1 and at HLT can change between luminosity blocks
  - A negative prescale means that this trigger is off. This is important for calculating the integrated luminosity. Check each luminosity block.

# Bringing it Together – TrigDecisionTool

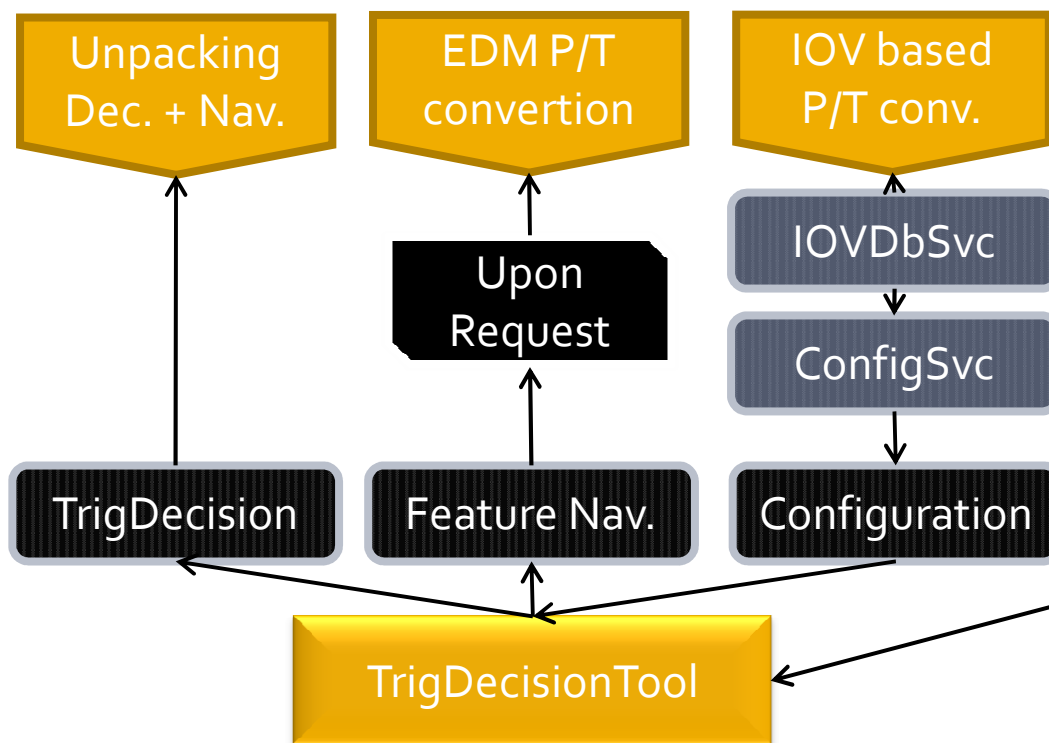


Configuration can change between runs

**BUT:**

Prescales can change between lumiblocks (trigger on/off)

Transient:



# Tool Usage

1. Alg.h: define ToolHandle to a TrigDecisionTool

```
private:  
    ToolHandle<Trig::TrigDecisionTool> tdt;
```

2. Alg.cxx – Alg::Alg(): declare ToolHandle as public tool

```
MyAlgo::MyAlgo(const std::string &name, ...)  
    tdt("Trig::TrigDecisionTool/TrigDecisionTool")  
    {...}
```

3. Alg.cxx – Alg::initialize(): retrieve tool

```
StatusCode sc = tdt.retrieve();
```

- Has to be in initialize()!

4. Alg.cxx – Alg::execute(): use tool

```
if (tdt->isPassed ("L2_e15i")) {  
    log << MSG::INFO << "I'm happy!" << endreq;  
}
```

TWIKI: <https://twiki.cern.ch/twiki/bin/view/Atlas/TrigDecisionTool15>

Doxygen: <http://atlas-computing.web.cern.ch/.../TrigDecisionTool/html/>

# Working with ChainGroups

## 1. Alg.h: define ChainGroup pointer

```
const Trig::ChainGroup* mMyTrigger;
```

## 2. Alg.cxx – Alg::initialize(): declare ChainGroup

- Note: the ChainGroup automatically updates with each run (chain content) and lumiblock (prescales)
- Use regular expressions, e.g. "EF\_e.\*"

```
mMyTrigger =  
    tdt.createChainGroup("EF_e10_loose");
```

## 3. Alg.cxx – Alg::execute():

- Access to trigger configuration
- Access to trigger decision
- Access to trigger objects

```
bool useLB = ! mMyTrigger->getListOfTriggers().empty();
```

```
bool myEvent = mMyTrigger.isPassed();
```

```
const Trig::FeatureContainer fc = mMyTrigger.features();  
const std::vector< Trig::Feature< TrigTau > > taus =  
    fc.get();
```

## Anonymous ChainGroups:

- Note that one can work without steps 1 and 2. In Alg::execute() define the triggers on the fly:
- Equally fast (TDT keeps ChainGroups)

```
string tr ("EF_e10_loose");  
bool useLB = ! tdt->getListOfTriggers().empty(tr);  
bool myEvent = tdt->isPassed(tr);  
const Trig::FeatureContainer fc = tdt.features(tr);
```



- L1 Items: name, prescale
  - For more detailed studies: version, CTP-Id, bunch group, thresholds and multiplicities
- HLT Chains: name, level, prescale, streams
  - For more detailed studies: version, counter, trigger elements and multiplicities
- Trigger Release, configuration keys (useful for lookup in the TriggerDB)
  - No access with the TDT, but checkConfigTrigger.py

# Configuration Access with the TDT

<code>std::vector&lt; std::string &gt;</code>	<code><a href="#">getListOfTriggers</a> (const <a href="#">Trig::ChainGroup</a> *chaingroup) const</code> <i>List of trigger names of the ChainGroup chaingroup</i>
<code>std::vector&lt; std::string &gt;</code>	<code><a href="#">getListOfStreams</a> (const <a href="#">Trig::ChainGroup</a> *chaingroup) const</code> <i>List of stream names of the ChainGroup chaingroup</i>
<code>std::vector&lt; std::vector&lt; std::string &gt; &gt;</code>	<code><a href="#">getListOfTriggerElements</a> (const <a href="#">Trig::ChainGroup</a> *chaingroup) const</code> <i>List of lists of TE names in ChainGroup chaingroup. Inner vectors are for a single trigger step, the length of those is the trigger multiplicity.</i>
<code>float</code>	<code><a href="#">getPrescale</a> (const <a href="#">Trig::ChainGroup</a> *chaingroup, unsigned int condition=<a href="#">TrigDefs::Physics</a>) const</code> <i>Prescale of the ChainGroup chaingroup.</i>

- All methods come with two flavors, argument:  
    const [Trig::ChainGroup](#) \*chaingroup  
    const std::string &triggerNames=".\*"   
Here the ChainGroup is created on the fly from the pattern 'triggerNames' (regexp - ".\*" means 'all')
- All functions are forwarded from ChainGroup, `tdt->fnc(chgr,...)` is equivalent to `chGr->fnc(...)`
- Prescales: For single triggers the prescale combined for all levels is returned. If a multi-trigger ChainGroup contains an unprescaled trigger, the return value is 1. It is 0 otherwise.

- Lvl1 decision after dead-time veto
- HLT decision, information for pass-through'ed and for resurrected triggers
- Bunch groups that fired
- For more detailed studies:
  - Lvl1 decision before prescale, before veto
  - Error codes for HLT-algorithms
  - Last step of chain execution

# Decision Access with the TDT

bool [isPassed](#) (const [Trig::ChainGroup](#) \*chaingroup, unsigned int condition=[TrigDefs::Physics](#)) const

*true if given group of chains passed*

char [getBGCode](#) () const

*Get the bunch group code byte. BG X fired (X=0..7): `getBGCode() & (0x1<<X)`*

All trigger decisions through “isPassed()”, behavior depends on “condition” argument:

- **TrigDefs::Physics:** [default]
  - Pseudonym for **requireDecision | enforceLogicalFlow**, which means that both conditions are applied. Default for `isPassed()` and for `getPrescale()`.
- **TrigDefs::enforceLogicalFlow, TrigDefs::fullChain:**
  - combination of all three trigger levels.
- **TrigDefs::requireDecision:**
  - A decision must have been made, pass due to *pass-through* or *resurrection* is not enough.
- **TrigDefs::passedThrough:**
  - Event was passed through. A trigger chain can have a pass-through factor in order to record events independent of the decision. This is useful for understanding the trigger behavior. Restrict the `isPassed()` to those events, `requireDecision` must be off.
- **TrigDefs::allowResurrectedDecision:**
  - If a trigger is prescaled on a given event, that trigger chain is not executed. If that event is accepted it is often important to know also the trigger decision and features of the prescaled trigger (orthogonal triggers for efficiency determination.) For a fraction of these cases we execute the trigger and save its decision. This is accessed with `allowResurrectedDecision` (`requireDecision` must be off.)

- Access to features through the “TriggerNavigation”
- Features are created by FEX algorithms, they appear in StoreGate. A FEX also creates a “TriggerElement” (TE)
  - A TE is used as handle to the feature
  - A TE has a pass/fail state set by the corresponding HYPO
- The navigation gives you all the features that were created in a chain
  - Or just those that were successful (features in ROI's which passed all cuts) – that is the default!
  - Can also give you combinations of features (for multi-object triggers)
  - Ancestor function to navigate, e.g. from electron to track and cluster

# Feature Access with the TDT

const [FeatureContainer](#) [features](#) (const [ChainGroup](#) \*group, unsigned int condition=[TrigDefs::Physics](#)) const  
*returns all features related to given chain group*

template<class T> const [Feature](#)< T > [ancestor](#) (const [HLT::TriggerElement](#) \*te, std::string label="") const  
*gives back feature matching (by seeding relation) te - is trigger element to start with, note that thanks to conversion operators [Feature](#) object can be given here as well*

- condition == [TrigDefs::Physics](#): contains features where ROI passed the last HYPO
- condition != [TrigDefs::Physics](#): all features created in trigger

## [FeatureContainer](#):

template<class T> const std::vector  
< [Trig::Feature](#)< T > > [get](#) (const std::string &label="", unsigned int condition=[TrigDefs::Physics](#),  
const std::string &teName="") const  
*returns flattened vector of Features of given type This method is in fact sullied by 3 arguments.*

const std::vector  
< [Trig::Combination](#) > & [getCombinations](#) () const  
*returns reference to combinations collected through append*

## [Combination](#)

template<class T> const std::vector  
< [Trig::Feature](#)< T > > [get](#) (const std::string &label="", unsigned int condition=[TrigDefs::Physics](#),  
const std::string &teName="") const  
*Method used to get objects.*

# Working with Features

- Access always a two or three step process
  1. FeatureContainer *fc* from the TDT
  2. List of Features<T> from *fc*  
or
  2. List of Combinations from *fc* and 3. list of Features<T> from each Combination
- [Feature< T >](#) has access to
  - Its constant object: `const T*`
  - The TriggerElement to which the object has been attached
    - This can be used with `TDT::ancestor(TriggerElement*)` method to find predecessors
  - Label (key in SG)

## [Feature< T >](#):

[operator const T \\*](#) () const

*implicit conversion to object pointer (explicit conversion [cptr](#) () for python)*

[operator const HLT::TriggerElement \\*](#) () const

*implicit conversion to TriggerElement (explicit conversion [te](#) () for python)*

[operator const std::string](#) () const

*explicit conversion to feature label (explicit conversion [label](#) () for python)*

# Athena Example for using Features

FeatureContainer for  
'EF\_tau16i\_loose\_2j23'

Access the feature  
You need to know the  
type of the Feature:  
'JetCollection' in EF

Access the object  
(implicit conversion)

Access information of  
object

Find corresponding L2  
jet using  
TDT::ancestor<T>

```
FeatureContainer f = tdt->features("EF_tau16i_loose_2j23"); // creating the feature container
std::vector< Feature<JetCollection> > jetColls = f.get<JetCollection>();
mLog << MSG::INFO << "Number of JetCollections: " << jetColls.size() << endreq;
if(jetColls.size(>0) {
    const Feature<JetCollection>& jcf = jetColls[0]; // get the first Feature
    mLog << MSG::INFO << "Label: " << jcf.label() << endreq;
    const JetCollection* jc = jcf; // implicit Feature -> object conversion
    mLog << MSG::INFO << "Number of Jets: " << jc->size() << endreq;
    JetCollection::const_iterator jlt = jc->begin();
    for (; jlt != jc->end(); ++jlt ) {
        Jet* jet = *jlt;
        mLog << MSG::INFO << "Jet e : " << jet->e() << endreq;
    }
    // find the corresponding jets in Lvl2 through the inheritance tree (navigation does that all)
    Feature<TrigT2Jet> l2jetF = tdt->ancestor<TrigT2Jet>(jcf);
    mLog << MSG::INFO << "Found " << (l2jetF.empty()?"no ":"") << "corresponding L2 Jet." << endreq;
    if ( !l2jetF.empty() ) {
        const TrigT2Jet* t2jet = l2jetF.cptr(); // explicit Feature -> object conversion
        mLog << MSG::INFO << " e : " << t2jet->e() << endreq;
    }
}
```

## Output:

```
Number of JetCollections: 1
TE Label: TrigJetRec
Number of Jets: 1
Jet e : 82827.9
Found corresponding L2 Jet.
e : 83197.4
```



- 14.2.22+: ARA with trigger data works, but limited to configuration and trigger object access. TrigDecisionTool and Navigation don't work with ARA here.
- 15.3.0+: new TrigDecisionTool. Everything works with ARA and **like in Athena**.
  - Running over multiple files with different configurations works seamlessly
- Athena and ARA work both in C++ and in python
  - Note templated functions syntax:
    - C++: `fc.get<TrigRoiDescriptor>("initialRol")`
    - Python: `fc.get('TrigRoiDescriptor')('initialRol')`
  - Implicit conversions don't work in python
    - Use `Feature::cptr()` and `Feature::te()`
  - Examples for all 4 cases are in [Trigger/TrigAnalysis/TrigAnalysisExamples](#)

- **FEXs** create **trigger objects**, which they attach to a **TriggerElement**, sometimes with a **Label**.
  - Two examples for the current 10<sup>31</sup> menu and 15.4.0:
    - `InDet::Pixel_TrgClusterization/PixelClustering_Tau_EFID` attaches `PixelClusterContainer` objects to `EFtau16i_loosetr`.
    - `TrigJetRec/TrigJetRec_Cone` attaches `JetCollection` objects to `EF_j23`, this time with the extra label `"TrigJetRec"`
- Q: Which FEX creates what type of object and what label ?
  - A: One needs to know the trigger software, and read the *literature* or look at the code. Use LXR and the algorithm classname to find out what is produced.
  - This can be of help:
    - Looking at the **Trigger EDM TWIKI** (type, label, slice, LXR): <https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerEDM>
    - Checking the **AOD content** (which labels exist for which type in the file):
      - `checkFile.py AOD.pool.root`
    - I recommend looking at [TrigEventARA/selection.xml](#) for the type used in the navigation (the one you need)

# Trigger Object Labels

- Feature access methods in the TrigDecisionTool have an optional argument "label". These correspond to the StoreGate keys (minus "HLT\_").

```
> checkFile.py AOD.pool.root | grep TrigRoiDescriptor
```

```
52.334 kb      5.552 kb      0.022 kb      0.199      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_TrigCaloRinger
56.061 kb      7.773 kb      0.031 kb      0.186      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_secondaryRoI_EF
63.160 kb     10.673 kb     0.043 kb      0.164      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_forMS
63.160 kb     10.679 kb     0.043 kb      0.164      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_forID
66.552 kb     11.474 kb     0.046 kb      0.156      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_secondaryRoI_L2
96.685 kb     25.609 kb     0.102 kb      0.108      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_TrigT2CaloEgamma
96.325 kb     27.150 kb     0.109 kb      0.108      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_TrigT2CaloTau
110.456 kb    31.728 kb     0.127 kb      0.095      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_TrigT2CaloJet
→ 163.605 kb   38.949 kb     0.156 kb      0.065      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_initialRoI
217.361 kb    60.725 kb     0.243 kb      0.050      250 (B) TrigRoiDescriptorCollection_tlp1_HLT
240.863 kb    70.211 kb     0.281 kb      0.046      250 (B) TrigRoiDescriptorCollection_tlp1_HLT_T2TauFinal
```

Correct way to get the initial RoIDescriptor (build from LVL1 ROI):

```
vector< Feature<TrigRoiDescriptor> > roi = fc.get<TrigRoiDescriptor>("initialRoI");
```

Note: the persistent data the type is *TrigRoiDescriptorCollection*, while through the navigation you get *TrigRoiDescriptor* objects. Again, the best way to find out is, at the moment, to look at [TrigEventARA/selection.xml](#).

# Expert Access with the TDT

- Some methods are not needed for most physics analyses, but for detailed trigger investigation (rate tool, timing tools, trigger validation)
- Access via  

```
Trig::ExpertMethods* em = tdt->ExperimentalAndExpertMethods();  
em->enable();
```

## ExpertMethods:

```
const TrigConf::TriggerItem* getItemConfigurationDetails (const std::string &chain)
```

*return TrigConf::TriggerItem chain: name of the item*

```
const TrigConf::HLTChain* getChainConfigurationDetails (const std::string &chain)
```

*return TrigConf::HLTChain chain: name of the chain*

```
const LVL1CTP::Lvl1Item* getItemDetails (const std::string &chain) const
```

*return [LVL1CTP::Lvl1Item](#) chain: name of the item*

```
const HLT::Chain* getChainDetails (const std::string &chain) const
```

*return [HLT::Chain](#) chain: name of the chain*

```
const HLT::NavigationCore* getNavigation () const
```

*return [HLT::NavigationCore](#)*

Access to  
trigger  
configuration

Access to  
complete  
trigger  
decision

Access to  
Navigation

# Trigger Object Matching

- Why?
  - Some analyses require **object level** trigger information: Tag and Probe
  - Some analyses have **multiple objects** of same type and need matching for detailed efficiency estimation
  - Trigger selection design and optimisation
- Matching framework available in Trigger/TrigAnalysis/TrigObjectMatching package in release **15.3.0** and later
- Examples available in Trigger/TrigAnalysis/TrigAnalysisExamples in release **15.4.0** and later
  - **TrigMatchExAlg** - **Athena** example of matching for each common offline type
  - **TrigMatchExampleARA** - Example of trigger matching in **ARA**
  - **TagAndProbeExAlg** - Athena example of **tag and probe** analysis with trigger matching
- **TWiki** available with detailed information for users and developers
  - <https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerObjectsMatching>

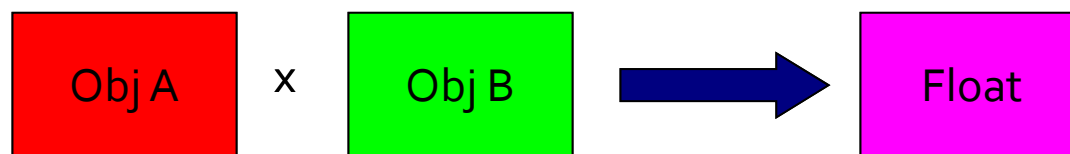
# Matching Framework Overview

- Two basic questions
  - What **chains** is the offline object associated with?
    - **chainPassedByObject** - does reco object match a passed trigger object in a given chain?
    - **chainsPassedByObject** - list of chains that reco object passes
  - What are the **properties** of the trigger object the offline object is associated with?
    - **matchToTriggerObjects** - vector of trigger objects matching to reco object in given chain
    - **matchToTriggerObject** - return best matching trigger object in given chain

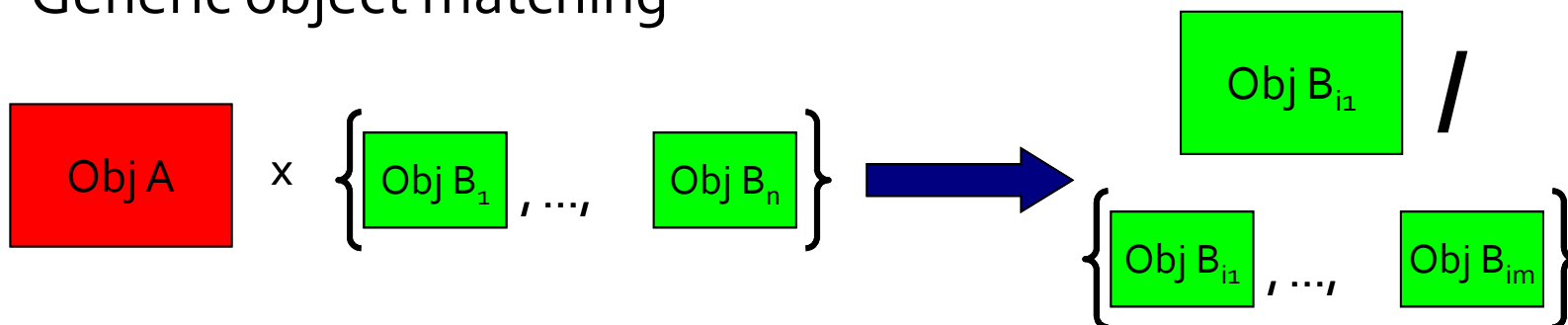
# Matching Framework Implementation

- Three basic problems:

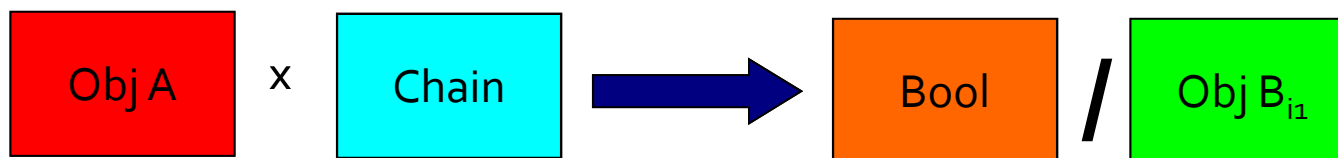
- Object distance



- Generic object matching



- Trigger specific matching



- Distance definition is implemented as a **function object** and passed as an **argument**
  - By default,  $\Delta R$  is used
  - Arbitrarily complex matching possible
- Quality of matches
  - Smallest  $\Delta R$  is **not a guarantee** of a good match
- Container type objects
  - EF trigger objects attached as **containers**, so matches are to a container rather than an individual trigger object
- What it means to pass a chain
  - Chain passed if match to trigger object associated with active trigger element
  - For multi-object triggers or intermediate trigger objects, an offline object can **pass a failed chain**
- Relative trigger level efficiencies
  - Possible to pass EF without passing L2 (passthrough, direction resolution effects)



- <http://trigconf.cern.ch>:
  - Main web portal to trigger configuration by run number, MC menu name, configuration keys
  - Uses TriggerTool and AtlCoolTrigger.py underneath
- TriggerTool:
  - Java based GUI to browse the TriggerDB (replica) for all information
  - Also used at point 1 for preparation of trigger
- AtlCoolTrigger.py:
  - Command line tool to show release, configuration keys, menu information, streams for data
- <http://atlas-runquery.cern.ch/>
  - Query page for trigger chains, prescales, trigger release for data

# Example for the Configuration Portal

- Go to <http://trigconf.cern.ch>

- Browse the trigger configuration (definition, algorithms, selection cuts)

Listing of Trigger Keys by Run

91000-91500

Example: 91000-92000,90275,93500-

1. Enter run-range

Listing trigger configurations

run  OR smk  L1 psk  and HLT psk

Specify either run number or set of configuration keys to display the trigger configuration

Comparing trigger configurations

run  OR smk  L1 psk  and HLT psk

Specify either run number or set of configuration keys for comparison with the trigger configuration above

2. Click on link in resulting run list

Also with simple comparison functionality

Click on L1 prescale to get to full trigger menu display. Mark two menus and click

run	Start Time	SMK	HLT PSK	L1 PSKs
91001	Wed Oct 8 15:28:35 2008	351	368	<a href="#">S20 (1-)</a> <input type="checkbox"/>
91003	Wed Oct 8 15:36:43 2008	351	368	<a href="#">S20 (1-)</a> <input type="checkbox"/>
91007	Wed Oct 8 15:54:39 2008	351	368	<a href="#">S20 (1-)</a> <input checked="" type="checkbox"/>
91043	Wed Oct 8 21:03:23 2008	355	373	<a href="#">S20 (1-)</a> <input checked="" type="checkbox"/>
91044	Wed Oct 8 21:27:12 2008	351	369	<a href="#">S20 (1-)</a> <input type="checkbox"/>
91045	Wed Oct 8 21:37:16 2008	351	369	<a href="#">S20 (1-)</a> <input type="checkbox"/>
91047	Wed Oct 8 22:04:03 2008	351	369	<a href="#">S20 (1-)</a> <input type="checkbox"/>
91056	Wed Oct 8 22:42:37 2008	351	369	<a href="#">S20 (1-)</a> <input type="checkbox"/>
91059	Wed Oct 8 23:57:22 2008	351	368	<a href="#">S20 (1)</a> <input type="checkbox"/> <a href="#">S20 (2)</a> <input type="checkbox"/> <a href="#">S21 (3)</a> <input type="checkbox"/>
91060	Thu Oct 9 01:28:35 2008	351	368	<a href="#">S20 (1)</a> <input type="checkbox"/> <a href="#">S22 (2-5)</a> <input type="checkbox"/> <a href="#">S20</a> <input type="checkbox"/>
91077	Thu Oct 9 12:25:35 2008	351	368	<a href="#">S15 (1-)</a> <input type="checkbox"/>
91086	Thu Oct 9 13:12:05 2008	351	368	<a href="#">S15 (1-2)</a> <input type="checkbox"/> <a href="#">S9 (3-)</a> <input type="checkbox"/>
91112	Thu Oct 9 14:13:32 2008	357	356	<a href="#">S19 (1-)</a> <input type="checkbox"/>

SMK 369 HLT Prescales Key: 390 Lvl1 Prescales Key: 541

If more advanced browsing is needed please launch [TriggerTool](#) Public Web Kit. If you have trouble to launch it.

## Streams

[L1Calo](#) | [RNDM](#) | [TGCwBeam](#) | [MBTS\\_BCM\\_LUCID](#) | [RPCwBeam](#) | [CosmicMuons](#) | [IDCosmic](#) | [IDTracks](#) | [express](#) | [BPTX](#) | [L1CaloEM](#) | [CosmicDownwardMuons](#) | [Tile](#) | [LARCells](#) |

L1Calo					
EF chain	PS	PT	STP	L2 chain	PS
e10_loose	1	0	1	e10_loose	1
e10_loose_passL2	1	0	1	e10_loose	1
e10_loose_passEF	1	0	1	e10_loose_passL2	1
e10_medium	1	0	1	e10_loose_passEF	1
e10	1	0	1	e10_medium	1
fau12_loose	1	0	1	e10	1
fau161_loose	1	0	1	fau12_loose	1
fauNoCut	1	0	1	fau161_loose	1
I5	1	0	1	fauNoCut	1
I10	1	0	1	I5	1
I70	1	0	1	I10	1
B110	1	0	1	I70	1
FJ18	1	0	1	B110	1
FJ150	1	0	1	FJ18	1
xe20	1	0	1	FJ150	1
fau161_EFxe30	1	0	1	xe20	1
I50	1	0	1	fau161_loose	1
xe_debug_EFonty	1	0	1	I50	1
erk9i	1	0	1	I23	1
erk16i	1	0	1	erk9i	1
erk9i_id	1	0	1	erk16i	1
fauNoCut_TauRecNoTopo	1	0	1	erk9i_id	1
fauNoCut_calo	1	0	1	fauNoCut_TauRecNoTopo	1
e5_NoCut	1	0	1	fauNoCut_calo	1
e5_NoCut_IdScan	1	0	1	e5_NoCut	1
e5_NoCut_TRT	1	0	1	e5_NoCut_IdScan	1
e5_NoCut_SITrk	1	0	1	e5_NoCut_TRT	1
e5_NoCut_FwdBackTrk	1	0	1	e5_NoCut_SITrk	1
xe20_noMu	1	0	1	e5_NoCut_FwdBackTrk	1
e10_calib	1	0	1	xe20_noMu	1
xe20_noiseSupp	1	0	1	e10_calib	1
xe20_FEB	1	0	1	xe20_noiseSupp	1
e5_nocut	1	0	1	xe20_FEB	1
xe150_noMu	1	0	1	e5_nocut	1
SingleBeamL1Calo	1	0	1	xe150_noMu	1
				SingleBeamL1Calo	1

RNDM					
EF chain	PS	PT	STP	L2 chain	PS
MbSpTrk	1	0	1	MbSpTrk	1
MbRndm	1	0	1	MbRndm	1
MbSp	1	0	1	MbSp	1
SingleBeamRNDM	1	0	1	SingleBeamRNDM	1

TGCwBeam					
EF chain	PS	PT	STP	L2 chain	PS
MBTGC_SpFwd_pixsct	1	0	1	MbTGC_SpFwd_pixsct	1
MBTGC_Mixs_1	1	0	1	MbTGC_Mixs_1	1
MBTGC_Mixs_2	1	0	1	MbTGC_Mixs_2	1
MBTGC_SpFwd_sct	1	0	1	MbTGC_SpFwd_sct	1

- [Up](#)
- [Top](#)
- [Streams](#)
- [L1 Items](#)
- [L1 Thresholds](#)
- [L2 chains](#)
- [EF chains](#)
- [Groups](#)
- [Sequences](#)
- [Help/Feedback](#)

- Trigger Menu and L1 rates stored in COOL, HLT rates coming. Quick access via
  - AtlCoolTrigger.py (command line tool)
    - AtlCoolTrigger.py -r 91000-99000 (many run summary)
    - AtlCoolTrigger -v -m -r 90272 (single run menu)
      - Prints keys, trigger menus, streams, allows diff-ing of menus in different runs, creates menu in xml format
  - Run summary pages (WEB based): <http://atlas-service-db-runlist.web.cern.ch/atlas-service-db-runlist/query.html>
    - Trigger names, rates for single run

- Java based front end to TriggerDB, launch from the web (Java web-start):

<http://www.cern.ch/triggertool>

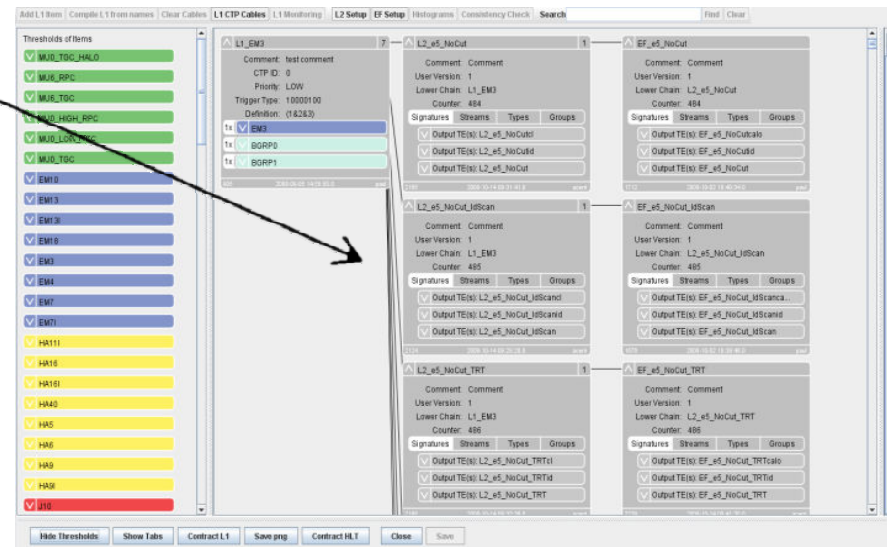
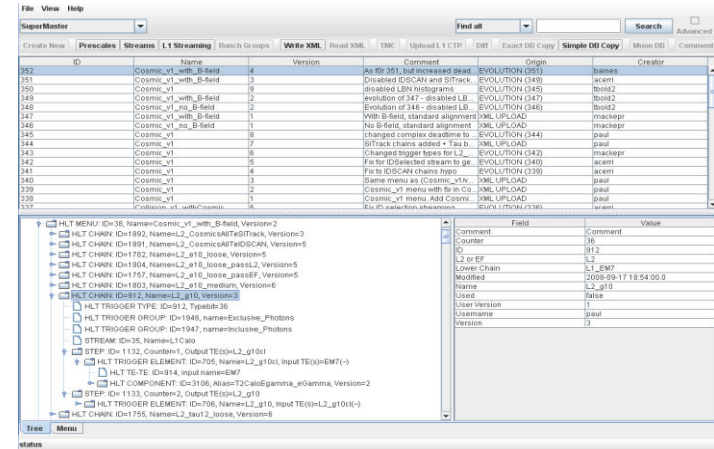
- Overview of all trigger configurations (data taking and MC)
- Detailed and convenient investigation of trigger menus

- Trigger definition L1->L2->EF: prescales, threshold algorithms, selection criteria, streaming information, etc.

- Possibility to compare different trigger configurations

- Used at point 1 for trigger operation

- [TDAQ training tutorial for the TT](#)



# Scripts to Check Pool File Content

- checkTrigger.py AOD.pool.root
  - Runs over ESD/AOD/DPD and presents detailed (chain-wise) counts of the trigger decision
- checkTriggerConfig.py -d AOD.pool.root
  - Runs on ESD/AOD/DPD and presents detailed trigger configuration(s)
  - Shows multiple configurations (merged DPD)

```
File:AOD.pool.root
Size: 55955.492 kb
Nbr Events: 250

Trigger configuration summary:
SMK: 0, HLTpsk: 0, L1psk: 0
Config source: TriggerMenuXML/LVL1config_MC_lumiE31_no_prescale_15.1.0.xml and
TriggerMenuXML/HLTconfig_MC_lumiE31_no_prescale_15.1.0.xml
L1 Items : 146
HLT Chains : 556
```

```
=====
ID level  Trigger name  Passed events: raw, after PS, after PT/Veto
=====
LVL1 Global LVL1          250
LVL2 Global LVL2          250
EF Global EF (LVL3)      250
-----
13 LVL1 L1_2EM13          71  71  71
14 LVL1 L1_2EM13l         34  34  34
163 LVL1 L1_2EM13_MU6     8   8   8
....
...
77 LVL2 L2_2g10           118 118 118
246 LVL2 L2_2g10_mu6     12  12  12
....
...
477 EF EF_2e6_medium      8   8   8
478 EF EF_2e6_medium1    7   7   7
79 EF EF_2g20             39  39  39
248 EF EF_2j42_xe30       3   3   3
....
=====
```

```
> checkTriggerConfig.py -d data09_cos.00121733.physics_L1Calo.recon.ESD.r733_tido73522/ESD.073522._000001.pool.root.1
...
EF: EF_muon_tgc_halo_IDSCAN (1.00), L2: L2_muon_tgc_halo_IDSCAN (1.00), L1: L1_MUon_TGC_HALO (1) [streams: TGCwBeam]
...
```

# Trigger Content of Atlas-Runquery

## ATLAS Run Queries

Run Summaries Trigger Configuration Query AMI Data Search DDM Dashboard Tier-0 Monitoring DQ Monitoring Data Preparation Operations

### Run Search - Insert Your Query:



[ default query condition ]

Examples (query format inspired by SPIRES):

**Hands-on session earlier today**

Run and event ranges Time ranges and duration Detectors Streams Magnets Data quality Project tag Trigger Partition

f r 90270-90300 / sh r and ev 100k+ / sh r and allev and smk and rel and tr EF\_e5\*

f r 91890-92070 / sh r and smk 368,373 / sh smk and rel and tr EF\_e5\*

f r 91890-92070 / sh tr L2\_E\*,L2\_Cosmic\* and rel and tr EF\_e5\*

f r 91890-92070 / sh r and tr EF\_e5\*

Search Results

Selection rule: f r 91890-92070 / sh r and allev and smk and rel and tr EF\_e5\*

Query command: AtIRunQuery.py --run "91890-92070" --show run --show events --show allevents --show smk --show release --show "trigger" --verbose --projecttag "data08",data09" --partition "ATLAS"

Selection sources: [Checking for runs in run range [91890, 92070]] 35 runs found  
[Checking if the elements in matches "data08",data09"] 37 runs found  
[Checking if the elements in matches "ATLAS"] 35 runs found

No. of runs selected: 70

Execution time: 19.6 sec

- Search for runs by release, configuration key, trigger content
- Display and have links to run-summary, AMI, trigconf.ch, e-log

Run	Links	#LB	#Events	#Events (SFO)	#L2A	SMK	HLT PSK	L1 PSK	Release	Trigger Chains
91890	RS, AMI, Trigger, ELOG	35	2,022,080	2,297,668	5,787,181	368	388	n.a.	14.2.23.2	EF_e5_NoCut (1I0), EF_e5_NoCut_IdScan (1I0), EF_e5_NoCut_TRT (1I0), EF_e5_NoCut_SiTrk (1I0), EF_e5_NoCut_FwdBackTrk (1I0)
91891	RS, AMI, Trigger, ELOG	24	n.a.	n.a.	n.a.	368	388	n.a.	14.2.23.2	EF_e5_NoCut (1I0), EF_e5_NoCut_IdScan (1I0), EF_e5_NoCut_TRT (1I0), EF_e5_NoCut_SiTrk (1I0), EF_e5_NoCut_FwdBackTrk (1I0)
91892	RS, AMI, Trigger, ELOG	1	2,344,840	2,620,933	6,743,545	n.a.	n.a.	n.a.	n.a.	
91893	RS, AMI, Trigger, ELOG	1	23,316	26,452	75,012	368	388	n.a.	14.2.23.2	EF_e5_NoCut (1I0), EF_e5_NoCut_IdScan (1I0), EF_e5_NoCut_TRT (1I0), EF_e5_NoCut_SiTrk (1I0), EF_e5_NoCut_FwdBackTrk (1I0)
91894	RS, AMI, Trigger, ELOG	1	0	0	0	368	388	n.a.	14.2.23.2	EF_e5_NoCut (1I0), EF_e5_NoCut_IdScan (1I0), EF_e5_NoCut_TRT (1I0), EF_e5_NoCut_SiTrk (1I0), EF_e5_NoCut_FwdBackTrk (1I0)
91895	RS, AMI, Trigger, ELOG	3	20	22	20	368	388	n.a.	14.2.23.2	EF_e5_NoCut (1I0), EF_e5_NoCut_IdScan (1I0), EF_e5_NoCut_TRT (1I0), EF_e5_NoCut_SiTrk (1I0), EF_e5_NoCut_FwdBackTrk (1I0)
91896	RS, AMI, Trigger, ELOG	2	40	46	40	368	388	n.a.	14.2.23.2	EF_e5_NoCut (1I0), EF_e5_NoCut_IdScan (1I0), EF_e5_NoCut_TRT (1I0), EF_e5_NoCut_SiTrk (1I0), EF_e5_NoCut_FwdBackTrk (1I0)

- Trigger data access in Athena, ARA, C++, or python with the TrigDecisionTool
  - Many examples, plus a large number of people to provide help when needed (hn-atlas-TriggerHelp at cern.ch)
- Tools to check the trigger information for given run
  - Configuration: TriggerTool or <http://trigconf.cern.ch>
  - Pool files: checkTrigger.py and checkTriggerConfig.py
- Prescales to be checked for each lumiblock → luminosity
  - Remember that negative prescale means trigger did not run
- Important for analysis: matching of offline reconstruction and trigger objects

# Additional Information

Trigger user info: Tutorials:	<a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerUserPages">https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerUserPages</a> <a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerSoftwareTutorialPage">https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerSoftwareTutorialPage</a>
TDT Twiki: TDT Doxygen:	<a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TrigDecisionTool">https://twiki.cern.ch/twiki/bin/view/Atlas/TrigDecisionTool</a> <a href="http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDevDirectory/AtlasOffline/latest_doxygen/InstallArea/doc/TrigDecisionTool/html/index.html">http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDevDirectory/AtlasOffline/latest_doxygen/InstallArea/doc/TrigDecisionTool/html/index.html</a>
Trigger obj matching:	<a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerObjectsMatching">https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerObjectsMatching</a>
TrigAnalysisExample:	<a href="http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDevDirectory/AtlasOffline/latest_doxygen/InstallArea/doc/TrigAnalysisExamples/html/index.html">http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDevDirectory/AtlasOffline/latest_doxygen/InstallArea/doc/TrigAnalysisExamples/html/index.html</a>
UserAnalysis example:	<a href="https://twiki.cern.ch/twiki/bin/view/AtlasProtected/UserAnalysis">https://twiki.cern.ch/twiki/bin/view/AtlasProtected/UserAnalysis</a>
Trigger Configuration: TriggerTool: Run query: Trigger EDM:	<a href="http://trigconf.cern.ch">http://trigconf.cern.ch</a> <a href="http://www.cern.ch/triggertool">http://www.cern.ch/triggertool</a> <a href="http://atlas-runquery.cern.ch/">http://atlas-runquery.cern.ch/</a> <a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerEDM">https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerEDM</a> , <a href="http://alxr.usatlas.bnl.gov/lxr/source/atlas/Trigger/TrigEvent/TrigEventARA/TrigEventARA/selection.xml">http://alxr.usatlas.bnl.gov/lxr/source/atlas/Trigger/TrigEvent/TrigEventARA/TrigEventARA/selection.xml</a>
TriggerMenu group: TriggerSW group: TriggerConfig group:	<a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerPhysicsMenu">https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerPhysicsMenu</a> <a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TAPMCoreSW">https://twiki.cern.ch/twiki/bin/view/Atlas/TAPMCoreSW</a> <a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerConfiguration">https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerConfiguration</a>
Help on e-groups:	<a href="https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerHelp">hn-atlas-TriggerHelp at cern.ch</a>