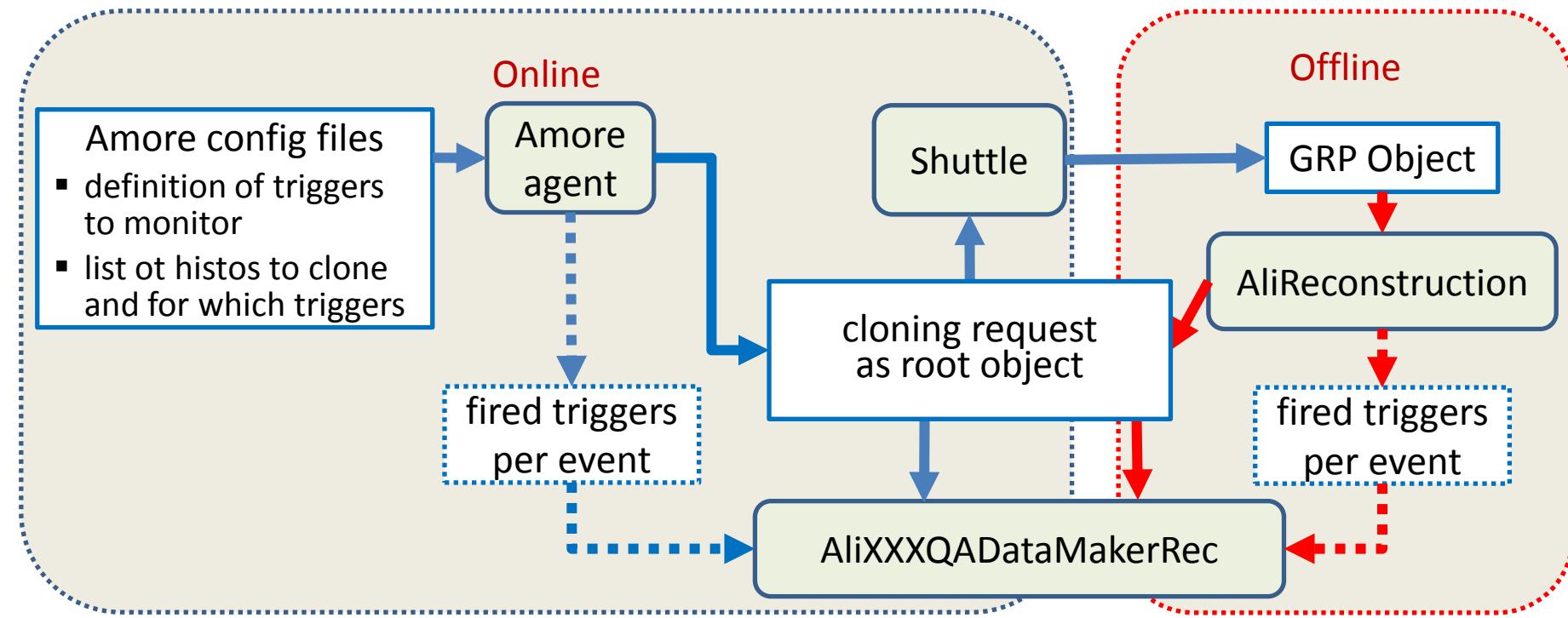


Extension of QA framework to optional trigger-specific histograms

Requested:

- ❑ possibility to produce in online DQM some histos separately for predefined trigger classes (or group of classes)
- ❑ repeat the same in the offline QA during reconstruction



Definition of triggers to monitor

```
INT          CINT7WU-B-NOPF-ALL  CINT7-B-NOPF-ALLNOTRD
EMCAL_ALL    CEMC7WU-B-NOPF-ALL
DIMUON_LS    CMUL7-B-NOPF-MUON
...
CMUS7-B-NOPF-MUON  CMUS7-B-NOPF-MUON
```

alias naming scheme still to be decided...

Definition of histograms to clone

```
VZERO/Raws/H1D_Trigger_Type INT EMCAL_ALL
VZERO/Raws/H2D_TimeV0A_V0C   INT DIMOUN_LS *
...

```

AliXXXQADataMaker:Init... :histogram booking

Procedure of histogram booking in InitXXX methods is not affected

```
AliQADataMaker::InitRaws {
```

```
    TH1* h1 = new TH1F(name1...);  
    Add2RawsList(h1, index1);  
    ...  
    TH1* h2 = new TH1F(name...);  
    Add2RawsList(h2, index2);
```

TObjArray* fRawsQAList[kNSpecies]							
<i>Event specie</i>	<i>histo index</i>						
kDefault							
kLowMultiplicity					<i>h1</i>		<i>h2</i>
kHighMultiplicity							

```
}
```

AliXXXQADataMaker:Init... :histogram booking, cloning

Procedure of histogram booking in InitXXX methods is not affected

Just need to call **ClonePerTriggerClass** method in the end of every InitXXX

This will substitute every histo requested to clone per specific triggers by TObjArray of cloned histos

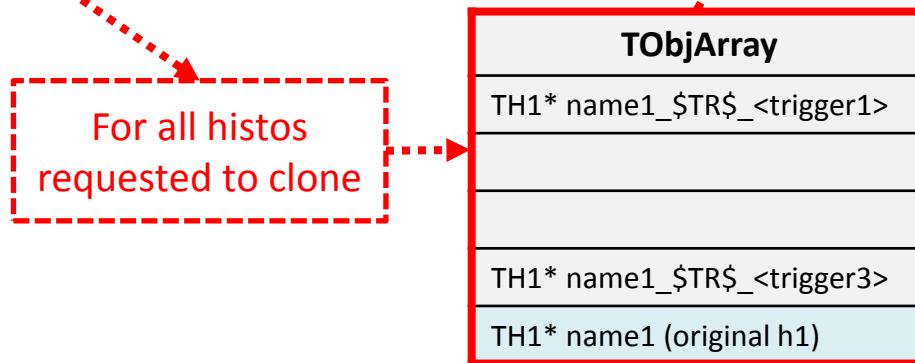
```
AliQADataMaker::InitRaws {
```

```
TH1* h1 = new TH1F(name1...);  
Add2RawsList(h1,index1);  
...  
TH1* h2 = new TH1F(name...);  
Add2RawsList(h2,index2);
```

TObjArray* fRawsQAList[kNSpecies]							
<i>Event specie</i>	<i>histo index</i>						
kDefault							
kLowMultiplicity					<i>h1</i>		<i>h2</i>
kHighMultiplicity							

```
ClonePerTrigClass(AliQAv1::kRAWS);
```

```
}
```



Attention: the old `TH1* AliQADataMaker::GetData(TObjArray**, int index)` and its aliases `GetRawsData...` now converted to `TObject* AliQADataMaker::GetData(TObjArray**, int index)` where returned TObject is either TH1 or TObjArray (test as `obj->TestBit(AliQAv1::GetClonedBit())==kTRUE`)

AliXXXQADataMaker::Make... histogram filling

In most of cases:

GetRawsData(id)->Fill(..) are substituted by `FillRawsData(id,...)`
same for GetRawsData(id)->SetBinContent(..) ⇒ `SetBinContentRawsData(id,...)` etc.
This will internally fill all clones of original histo as slot **id** matching to fired triggers of current event...

Different access methods for some non-standard cases:

`static const TObjArray& AliQADataMaker::GetEventTrigClasses()`
⇒ array of Tnamed's with monitored trigger classes of the event (GetName(): alias, GetTitle(): Alice trigger classes)
`TobjArray* AliQADataMaker::GetMatchingRawsData(int index, TobjArray* optDest)`
⇒ array of TH1's with clones matching to trigger classes of given event (or original histo if not cloned)
...

Handling of event counters

Some detectors had custom counters of effective MakeXXX calls, used later for normalization of histos.
Such normalizations will be wrong for histo clones filled for specific triggers!

Now one should use framework counters (per cycle and total);

They must be incremented by user in the MakeXXX as suitable (e.g. no increment if event was skept...)

`void IncEvCountCycle(AliQAv1::TASKINDEX_t task, Int_t diff=1);`
`void IncEvCountTotal(AliQAv1::TASKINDEX_t task, Int_t diff=1);`
+ aliases like `IncEvCountCycleRaws()`

This will increment event counter for all fired trigger classes + (general trigger-blinded counter, id=-1)

To access the counters:

`Int_t GetEvCountCycle(AliRecoParam::EventSpecie_t sp, AliQAv1::TASKINDEX_t task, Int_t trCl=-1)`
`Int_t GetEvCountTotal(AliRecoParam::EventSpecie_t sp, AliQAv1::TASKINDEX_t task, Int_t trCl=-1)`
+ some aliases for specific tasks, preset event species ...

AliXXXQADataMaker:EndOfDetectorCycle processing

In general, on top of the loop over species, the loop over all trigger classes is added:

```
for (Int_t specie=0; specie<AliRecoParam::kNSpecies; specie++) {  
    if (! IsValidEventSpecie(specie, list)) continue;  
    SetEventSpecie(AliRecoParam::ConvertIndex(specie));  
    for (int itc=-1;itc<GetNTrigClasses();itc++) { // -1 is for trigger-blind histos  
        float nEvent = GetEvCountCycleRaws(itc);  
  
        // to access specific histo (clone) at some slot  
        if (nEvent>0) if ( (htmp=GetRawsData(169,itc)) ) htmp->Scale(1./nEvent);  
  
        // or, to access all histos defined for monitored trigger its  
        TObjArray& arr = *GetRawsDataOfTrigClass(itc);  
        if (arr[169] && nEvent>0) ((TH1*)arr[169])->Scale(1./nEvent);  
        . . .  
    } // triggers  
} // ev.species
```

Some classes perform manipulation on a group of histograms.

I've modified these parts in such a way, that the operation is performed on the histograms for the same trigger (if the cloning was requested).

AliQAchecker

Select histos cloned for specific monitored triggers (or trigger-blind histos) and supplies them to detector QAcheckerBase as before:

```
void AliQAcheckerBase::Run(AliQAv1::ALITASK_t index, TObjArray ** list, . . .
{
    TObjArray ** listTrig = new TObjArray *[AliRecoParam::kNSpecies];
    for (int itc=-1;itc<AliQADataMaker::GetNTrigClasses();itc++) { // -1 for trigger-blind
        for (int specie=0;specie<AliRecoParam::kNSpecies;specie++) {
            listTrig[specie] = 0;
            if ( !AliQAv1::Instance()->IsEventSpecieSet(specie) || !list[specie]) continue;
            listTrig[specie] = new TObjArray( list[specie]->GetSize() );
            AliQADataMaker::GetDataOfTrigClass(list[specie],itc, listTrig[specie]);
        }
    }
    //Check(rv, index, list, recoParam); // Old code
    Check(rv, index, listTrig, recoParam);
    SetQA(index, rv);
    for (int specie=0;specie<AliRecoParam::kNSpecies;specie++)
        if (listTrig[specie]) delete listTrig[specie]; // clean temporary container }
    delete [] listTrig;
    Finish();
}
```

Comparison to QA reference histos from CDB:

1st look for the reference histo with “cloned histo name” <name>_STR\$<trigger_alias>
If not found, use the reference with “uncloned” <name>

Special cases

TPC

Accumulated data not in the framework histos but in custom object AliTPCdataQA

Not compatible with cloning



Solved (?): AliTPCdataQA was modified (P.Christiansen) to use histos provided by QA framework.

MUON

Uses custom objects (non TH1 based) for QA

Need more discussion/decisions on how to make them compatible with cloning

In any case, no problem is expected if no cloning request is done for detector which is not ready for it (modifications are backward compatible)

Note: if some histo should never be cloned, detector code may explicitly state this by calling

`AliQADataMaker::ForbidCloning(TH1* h, Bool_t v=kTRUE)`

in its InitXXX methode before the ClonePerTriggerClass is called.