# Feature2Wrapper:: PostUsecaseCreated

## CreateFeatureGraphManager

streamConfig.numStreams = numStreams; ( 3RDI+3FD+3拍照stream )
streamConfig.operationMode = m_pFrameworkStreamConfig->operationMode;
streamConfig.pChiStreams = &pStreams[0]; ( 3RDI+3FD+3拍照stream )
streamConfig.pSessionSettings = NULL;
streamConfig.pSessionSettings = m_pFrameworkStreamConfig->pSessionSettings;

fgmConfig.pCameraStreamConfig = &streamConfig;
fgmConfig.pCameraInfo = m_pLogicalCameraInfo;
fgmConfig.pSessionParams = 0;
fgmConfig.pMetadataManager = m_pMetadataManager;

featureWrapperCallbacks.ProcessCaptureResultCbToUsecase = ProcessCaptureResultCb;
featureWrapperCallbacks.NotifyMessageToUsecase = ProcessMessageCb;
featureWrapperCallbacks.ProcessPartialCaptureResultCbToUsecase = ProcessPartialCaptureResultCb;
featureWrapperCallbacks.pPrivateCallbackData = this;

### ChiFeature2GraphManager::Create

#### ChiFeature2GraphManager::Create

- CloneStreamConfig
- RegisterFGMCallbacks
- m_featureCallbacks.ChiFeature2ProcessMessage = ChiFeature2Graph::ProcessMessage
- CreateFeatureGraphSelector
  - LoadFeatureGraphSelectorOps ( 填充m_feature2GraphSelectorOps )
    - GetFeature2LibPath
    - funcPChiFeature2FGSOpsEntry = reinterpret_cast<PCHIFEATURE2GRAPHSELECTOROPSENTRY>( ChxUtils::LibGetAddr(handle, "ChiFeature2GraphSelectorOpsEntry"))

```
VOID ChiFeature2GraphSelectorOpsEntry(
    CHIFEATURE2GRAPHSELECTOROPS* pChiFeature2GraphSelectorOps)
{
    if (NULL != pChiFeature2GraphSelectorOps)
    {
        pChiFeature2GraphSelectorOps->size              = sizeof(CHIFEATURE2GRAPHSELECTOROPS);
        pChiFeature2GraphSelectorOps->pCreate           = CreateFeatureGraphSelector;
        pChiFeature2GraphSelectorOps->pGetFGDListForConfig = GetFeatureGraphMapforConfig;
        pChiFeature2GraphSelectorOps->pSelectFGD        = SelectFeatureGraphforRequest;
        pChiFeature2GraphSelectorOps->pAddCustomHints   = AddCustomFeatureGraphNodeHints;
        pChiFeature2GraphSelectorOps->pQueryCaps        = DoQueryCaps;
        pChiFeature2GraphSelectorOps->pDestroy          = Destroy;
    }
}
```

    - funcPChiFeature2FGSOpsEntry
  - m_feature2GraphSelectorOps.pCreate ChiFeature2GraphSelectorOEM::Create
    - pFeatureGraphSelector->PopulateAllTablesOEM
      - BuildCameraIdSet
        - pLogicalCameraInfo[i].logicalCameraType
          - LogicalCameraType_Default cameraIdSetSingle.insert(pLogicalCameraInfo[i].cameraId);
          - case LogicalCameraType_RTB:<br>case LogicalCameraType_VR:<br>case LogicalCameraType_SAT:<br>case LogicalCameraType_BAYERMONO:<br>case LogicalCameraType_DualApp:
            - LogicalCameraType_SAT cameraIdSetFusion.insert(pLogicalCameraInfo[i].cameraId)
            - LogicalCameraType_RTB cameraIdSetBokeh.insert(pLogicalCameraInfo[i].cameraId);
            - cameraIdSetMulti.insert(pLogicalCameraInfo[i].cameraId);
        - m_cameraIdMap.insert({ { cameraIdSetSingle }, SINGLE_CAMERA });<br>m_cameraIdMap.insert({ { cameraIdSetFusion }, FUSION_CAMERA });<br>m_cameraIdMap.insert({ { cameraIdSetBokeh }, BOKEH_CAMERA });<br>m_cameraIdMap.insert({ { cameraIdSetMulti }, MULTI_CAMERA });
      - pGraphDescriptorTables = GetGraphDescriptorTables
      - pGraphDescriptorTables->pCameraIdDescriptorNameSet
      - pGraphDescriptorTables->pFeatureGraphDescriptorsMap
      - pGraphDescriptorTables->pFeatureGraphDescKeysMap
      - m_featureDescNameSet = {, , , , , }
      - PopulateAllTables
    - pFeatureGraphSelector->m_featureDescNameSet
    - pFeatureGraphSelector->Initialize
- ChiFeaturePoolManager::Create ChiFeaturePoolManager::Initialize
  - ChiFeaturePoolManager::ProbeChiFeature2Features ( 将/vendor/lib64/下面的feature so的capacity和m_featureDescNameSet作比较，如果匹配，则insert到m_featureNameToOpsMap，这个map就是有效的feature的map )
    - funcPChiFeature2OpsEntry = reinterpret_cast<PCHIFEATURE2OPSENTRY>(ChxUtils::LibGetAddr(handle, "ChiFeature2OpsEntry"))
    - funcPChiFeature2OpsEntry(&feature2Ops)
    - feature2Ops.pQueryCaps(m_pConfig, &caps)
    - pCapability = caps.ppCapabilities

```
if ((NULL != pCapability))
{
    for (auto &it : m_featureDescNameSet)
    {
        if (!CdkUtils::StrCmp(it, pCapability))
        {
            m_featureNameToOpsMap.insert({ pCapability, feature2Ops });
        }
    }
}
```

- GetListofFeaturesSupported
- pGetFGDListForConfig ChiFeature2GraphSelector::GetFeatureGraphMapforConfig
  - GetAllFeatureGraphDescriptors
    - FALSE == featureGraphDesc.isMultiCameraGraph
    - TRUE == featureGraphDesc.isMultiCameraGraph

```
ChiFeature2FGDKeysForClonedMap phyCamDesc;
phyCamDesc.pDescriptorName   = descriptorName;
phyCamDesc.cameraId          = m_pCameraInfo->cameraId;
phyCamDesc.bMultiCamera      = TRUE;
phyCamDesc.featureFlags.value = 0;
```

      - CallCloneGraphDescriptor
        - m_clonedFeatureGraphDescriptorsMap.insert
        - m_pFeature2GraphDescs.push_back
      - mc暂时只有三个featureGraph （都在m_clonedFeatureGraphDescriptorsMap和m_pFeature2GraphDescs）
        - MultiCameraRawHDRIRaw2YUVGraph
        - MultiCameraMFNRFusionSuperGraph
        - MultiCameraRawMFSRIRaw2YUVGraph
  - rSelectorOutput.numFeatureGraphDescriptors = m_clonedFeatureGraphDescriptorsMap.size()
- m_pFeatureGraphDescMapForConfig （从map里面依次取出来featureGraph 然后Create，SAT要创建的有三个 一个默认，一个HDR 一个SR ）
  - m_pFeaturePoolManager->CreateFeatureInstance ChiFeaturePoolManager::CreateFeatureInstance
    - MultiCameraRawMFSRIRaw2YUVGraph
      - AnchorSync
        - DoStreamNegotiation chifeature2anchorsync.cpp DoStreamNegotiation 对于anchor来说 一个RDI和 一个FD输入， 一个RDI输出
          - GetSensorOutputDimension 获取每个phy Cam的RDI Size，
          - 子主题 2
            - pNegotiationOutput->pStreams->push_back(pRDIInputStream)
            - pNegotiationOutput->pStreams->push_back(pFDInputStream)
            - pNegotiationOutput->pStreams->push_back(pRDIOutputStream)
          - 子主题 3
            - pNegotiationOutput->pInputPortMap->push_back(std::pair<UINT8, UINT8>{portId, camIdx}) ( cameraID 0,0,1,2 CameraID 1 :3, 4,5 CameraID 2:6,7,8 )
            - pNegotiationOutput->pOutputPortMap->push_back(std::pair<UINT8, UINT8>{portId, camIdx})
        - UpdateInstancePropsInGraphDesc
        - DoFeatureCreate
        - m_pFeatureList.push_back(pChiFeature2Base);<br>m_pEnabledFeatures.push_back(pChiFeature2Base);<br>m_pEnabledFeaturesMap.insert({ feature2InstanceId, pChiFeature2Base })
      - FormatConvertor
      - RawMFSR
      - IRaw2Yuv
    - MultiCameraRawHDRIRaw2YUVGraph
    - MultiCameraMFNRFusionSuperGraph
- 子主题 9

## CreateTargetBufferManagers

- m_internalInputStreamMap(3RDI+3FD)
  - pChiTargetBufferManager = CHITargetBufferManager::Create
  - m_frameworkTargetBufferManagers[ pChiStream] = pChiTargetBufferManager
- m_pFrameworkStreamConfig(3个camera的拍照stream)
  - pChiTargetBufferManager = CHITargetBufferManager::Create
  - m_frameworkTargetBufferManagers[m_pFrameworkStreamConfig->pChiStreams[i]] = pChiTargetBufferManager
- m_pLogicalCameraInfo->numPhysicalCameras ( 三个camera的inputMetadata )
  - pTargetBufferManager = CHITargetBufferManager::Create
  - m_pInputMetadataTBMData.push_back( pTargetBufferManager)