Qualcomm

Qualcomm Technologies, Inc.

QACT v8.1 User Guide

80-VM407-21 Rev. AD

July 27, 2023

For additional information or to submit technical questions, go to: https://createpoint.qti.qualcomm.com

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
AA	March 2023	Initial release
AB	March 2023	Updated Section2 4.4, 4.7, and 4.8
AC	April 2023	Updated Section 4.3.5
AD	July 2023	Added Chapters 7 and 8. Updated Chapter 4.

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

Contents

1 Introduction

1.1 Purpose

This document describes how to use the QACT™ platform to configure and calibrate the voice and audio use cases and features available in the AudioReach™ software architecture. Since audio calibration requires the cooperation of multiple functional teams, this guide is organized according to engineering roles/workflows involved in each phase of calibration:

- System Designer – This role establishes the required infrastructure
- Tuning Engineer – This role tunes the audio path to pass product expectations

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `cp armcc armcpp`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

Shading indicates content that has been added or changed in this revision of the document.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

Part 1: Using QACT

2 Getting started

This chapter describes the system requirements for QACT and provides installation instructions.

2.1 System requirements

- Operating system – Windows 10
- Monitor – 1920 x 1080 (or 3840 x 2160 recommended)
- Microsoft .NET 4.6 framework – The Microsoft .NET framework version is verified during installation. If .NET 4.6 is not detected, the installation application indicates the necessary requirements and aborts the installation.

2.2 Verify QACT version compatibility

QACT 8.1 supports software products (SP) that contain the AudioReach audio software architecture. With QACT v8.1, use cases can be automatically created based on connected subgraphs.

QACT 8.0 supports SPs that contain the AudioReach audio software architecture. With QACT v8.0, users need to manually select subgraphs to design use cases.

QACT v7.4 and prior versions support SPs that contain the Elite audio software architecture.

To determine if a given target and software release are using AudioReach vs. Elite architecture, confirm the ACDB file version. The simplest way to do this is to perform a search on the software build files for *.acdb and/or *.qwsp files.

2.3 Qualcomm Package Manager

In order to support a wider customer base and business requirements and improve installation user experience, QACT has adopted the Qualcomm Package Manager (QPM). During the installation process, QPM will confirm the licenses assigned and automatically install necessary AddOns or Plugins.

The following table describes the capabilities provided in QACT_BASE and QACT_PLUGINs.

Feature/AddOn	License required	Tool capability
Core	▪ AMSS license ▪ PKLA license ▪ HAP license	▪ Full project configurator capability ▪ Customer topology workflow ▪ Full tuning capability (offline, online, RTC) ▪ License-free QTI modules (gain, IIR, FIR, MBDRC, etc.)
WCD_WSA	AMSS license	▪ Active noise cancellation (ANC) – Headset ▪ Speaker protection (SP) ▪ Microphone activity detection (MAD)
Headset	PKLA license	▪ QCSR/voice and music personal audio ANC
Advanced_Voice	Fluence Pro license	▪ Fluence Pro multi-mic echo cancellation and noise suppression (ECNS) ▪ Source tracking ▪ Sound focus
Essential_Voice	▪ Fluence license ▪ Fluence HD license ▪ Fluence Pro license	▪ Fluence dual-mic ECNS ▪ Widevoice ▪ Slowtalk ▪ Adaptive active noise cancellation (AANC) (handset) ▪ Far-end noise suppression (FENS and/or FNS)
Voice_UI	▪ Fluence HD license ▪ Fluence Pro license	▪ Qualcomm® Snapdragon™ Voice Activation (SVA) ▪ Single-mic barge-in – Linear echo cancellation (LEC)
Voice_Speech_Enh	FluencePro license	▪ Multi-channel (includes stereo EC) echo cancellation (MEC) ▪ Far-field voice (FFV) ▪ Sound focus ▪ Source tracking
Third_Party	AMSS license	Third party algorithms distributed by Qualcomm (Google, Dolby, DTS)
Other	Fluence Pro license	▪ Audiosphere ▪ Ambisonics ▪ Surround sound record ▪ Audio zoom

FAQ

How do I request QTI licensing for audio packages?

Contact your sales account manager to request a license for the audio packages.

I have a license but am not able to have the proper AddOn installed?

Raise a Salesforce case and request the visibility of the appropriate QACT_PLUGIN_* tool distribution. The case should be associated with:

- Problem Area 1 – Multimedia
- Problem Area 2 – Audio and Voice Tuning
- Problem Area 3 – Any

Include the QPM installer log file provided at the end of the installation process as seen in the following image.

Where did QACT Lite go?

QACT Lite has been replaced by the Core QACT license.

2.4 Install QACT

1. Uninstall any previous versions of QACT.
2. Ensure that the required .NET framework version is installed.
3. Open Qualcomm Package Manager.
4. Switch to the All tab, find Qualcomm Audio Calibration Tool, and click **Install**.
5. In the Welcome dialog, click **Next**.
6. Read the Software License Agreement and, if you agree, click **I Agree** to continue.
7. Click **Install**.
8. Click **Finish** to complete the installation.

2.5 Install and confirm QACT dependencies

2.5.1 SDM/QCS products

For SDM-based products, QACT is dependent on QPST or QUTS and the USB driver to perform connectivity features.

In 2019, QUTS replaced QPST to provide PC-based phone connectivity services. Going forward, it is recommended to use QUTS for connectivity.

2.5.1.1 QUTS setup

2.5.1.1.1 Install QUTS and the USB driver

1. Open Qualcomm Package Manager.
2. Switch to the All tab, find QUTS, and click **Install**.
3. Install the USB driver to enable the virtual COM ports through the USB.

2.5.1.1.2 Connect the target device in QUTS

NOTE: QUTS and QACT must be run in the same mode (User or Admin). If the modes of both applications do not match, QACT will be unable to connect to QUTS.

NOTE: Running QUTS in Admin mode is not recommended as it has the side effect of changing file ownership to Admin.

1. Launch QUTSStatus, which is an EXE tool installed together with QUTS.
2. Connected devices should be automatically shown in the Devices list.

2.5.1.2 QPST setup

2.5.1.2.1 Install QPST and the USB driver

3. Install and set up QPST. See *Qualcomm Product Support Tool (QPST™) 2.7 User Guide* (80-V1400-3) for instructions on how to set up the QPST connection.
4. Install the USB driver to enable the virtual COM ports through the USB.
5. Ensure that the ActiveSync Microsoft driver or equivalent is installed. This driver is required for QPST to communicate with a target/customer device.
6. Click **Start -> Programs -> QPST -> QPST Configuration**.
7. On the IP Server tab, click the **Accept client connections** checkbox.

2.5.1.2.2 Connect the target device in QPST

NOTE: QPST and QACT must be run in the same mode (User or Admin). If the modes of both applications do not match, QACT will be unable to connect to QPST.

NOTE: Running QPST in Admin mode is not recommended as it has the side effect of changing file ownership to Admin.

1. With the QPST Configuration application running, connect the target device to the QACT workstation via a USB cable.
2. Power on the target device.
3. On the QPST Ports tab, click **Add New Port**.
4. In the Add New Port dialog box, perform the following steps:
   a. Select the applicable USB port.
   b. Verify the **Port** and **Port Label** values.
   c. Select the Show Serial and USB/QC Diagnostic ports only checkbox.
   d. Click **OK**.
5. Click the **Ports** tab to see the new port.
6. Click the **Active Phones** tab to verify that the device has been detected. The **Active Phones** tab displays all detected devices.
7. Perform the following steps to ensure that the Embedded File System (EFS) of the target is viewable.
   a. Click Start Clients > EFS Explorer.
   b. In the Phone Selection dialog box, select the device that is currently connected. Click **OK**.

2.5.2 WCD/WSA EVB products

For WCD/WSA-based EVB products, QACT is dependent on the QTI Thesycon USB driver to be installed.

To install the QTI Thesycon USB driver, in the QACT installation directory, run and install <Installdir>\USBDrivers\ Qualcomm_USBAudio_v4.40.0_2018-01-29_setup.exe.

2.6 QACT Launcher View

Launcher View is the common first window that appears when QACT is started. It provides a way to open QWSP/ACDB files or connect to devices. Beyond this window, QACT's capabilities and presentation depends on what product is connected.

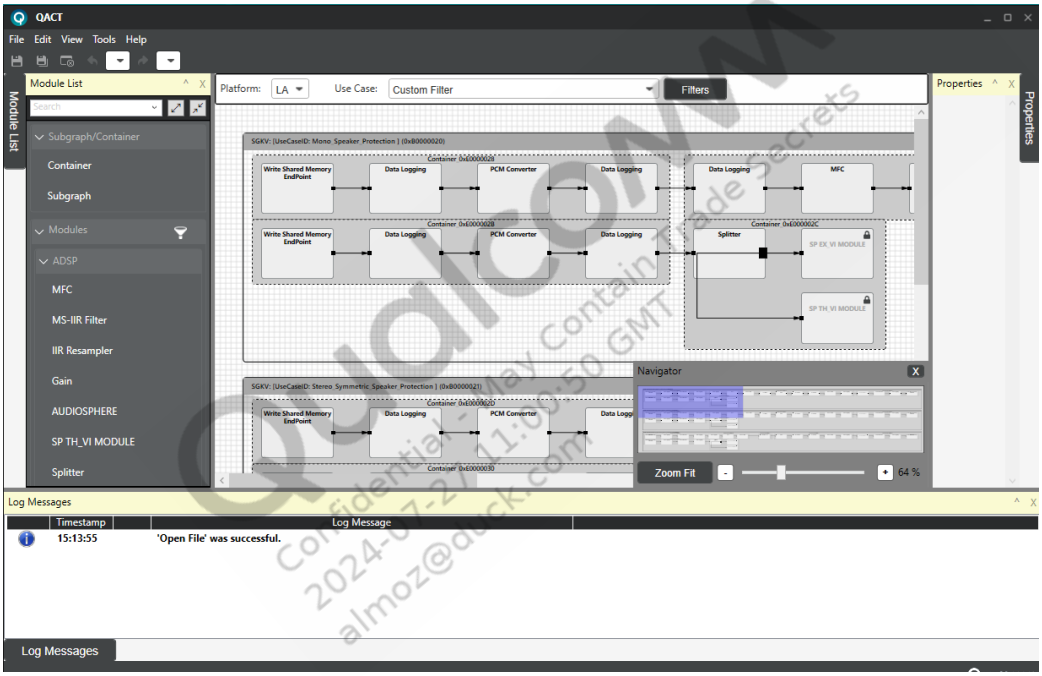
Launcher view provides the following capabilities:

Recently Opened Files	Lists the recently opened QWSP/ACDB files as a shortcut to open the files.
Recently Connected Devices	Lists the recently connected devices as a shortcut to connect to the device again.
Open ACDB File	Provides a way to browse and open a QWSP/ACDB file.
Connect to Device	Provides a way to detect and connect to connected devices.
Connection Configuration	Provides a mechanism to configure how QACT will connect to a device.
Diff/Merge	Provide a mechanism to compare and merge two sets of QWSP/ACDB files.
Create or Apply ACDB patch files	Create patch files by comparing two different ACDB files. Apply changes in patch files to a target ACDB file.
Release Notes	Opens the release notes for this tool version.
User Guide(s)	Opens this User Guide from Createpoint.

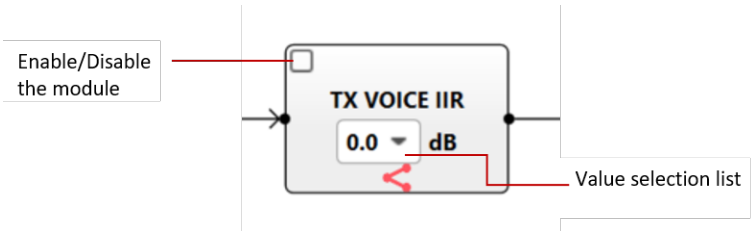
Part 2: SDM/QCC products

3 Using QACT with SDM/QCC products

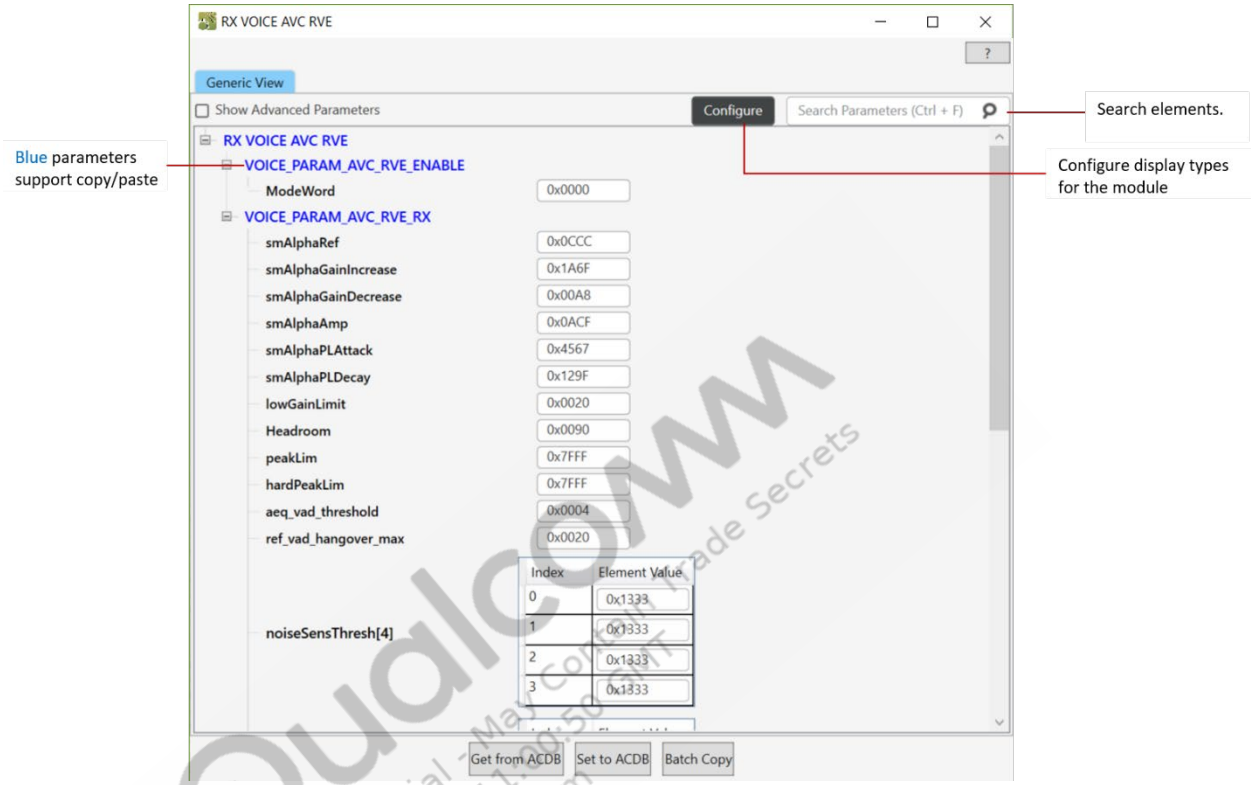
After a QWSP/ACDB file or device is connected, Canvas View appears. This view provides the user the ability to navigate product use cases and configure/tune the contents of QWSP/ACDB files.



Tuning modules can be enabled or disabled in Topology View. When applicable, calibration values can be selected from the list displayed on the module.



Double clicking a module in Topology View opens the module calibration dialog. Parameters displayed in blue font support copy and paste functionality.

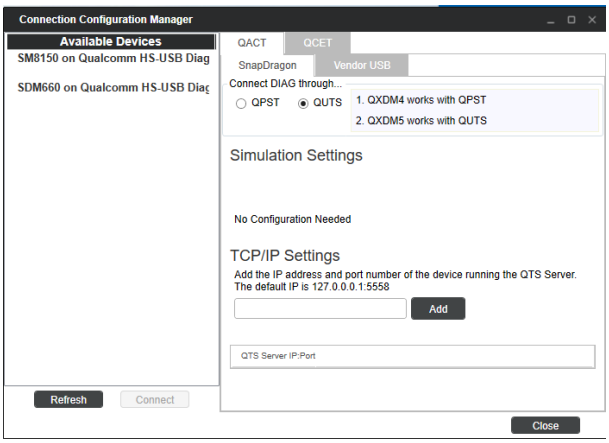


3.1 Connect to a target device

3.1.1 Configure connection

QACT can be connected using Qualcomm® Product Support Tool (QPST) or Qualcomm® Unified Tool Service (QUTS).

QPST/QUTS can be selected from Connection Configuration Manager (QACT launcher -> Connection Configuration) as shown.



QPST/QUTS are mutually exclusive and cannot be used simultaneously.

- If QUTS is selected, ensure that QPST is completely closed. Close all tools (for example, QXDM 4) that use QPST and check if QPST is running by looking in the system tray for the QPST icon.
- If QPST is selected, ensure that no other tool (for example, QXDM 5) is connected using QUTS.

3.1.2 Connect to an SDM product

1. If you are using an Android N build, run the following commands to give the audio server privileges to use the DIAG service (for other builds, skip this step):

```
adb root
adb remount
adb shell
chmod 777 /dev/diag
chmod 777 /dev/msm_rtac
chmod 777 /dev/msm_audio_cal
stop audioserver
start audioserver
```

2. Click **Connect to Phone**.
3. If two target devices have been connected, the Connecting dialog displays both target devices. In the Connecting dialog, select the applicable target and click **Connect**. If only one target device is connected, the Connecting dialog does not display; proceed to step 4.
4. The device list and parameter list are populated in QACT when a target device is connected. The status bar indicates whether the target is connected. Configuration parameters in targets that are connected to the QACT workstation can be modified using either of the following modes:
   - Online mode – See Section 5.1.2 to save customized parameters to an .acdb file.
   - Real-Time Calibration (RTC) mode – See Section 5.1.3 to write customized parameters directly to the DSP on the target device.
5. To close a connection to a target, click **File > Close**.

3.1.3 Connect to a QCS product

To connect to a QCS product:

1. Check for pulse audio presence:

```
# ps -ef | grep pulse
```

2. Stop pulse audio:

```
adb shell systemctl stop pulseaudio
```

3. Give diag permission:

```
adb shell chmod 777 /dev/diag
adb shell chmod 0666 /dev/ion
```

4. Restart pulse audio:

```
adb shell pulseaudio --system -vvv
```

5. Run any use case (for example, layback, SVA, or record) from the device. For example:

```
adb shell paplay -p <path>/<file name>.wav -d offload0
```

6. Click **Connect to Device.**

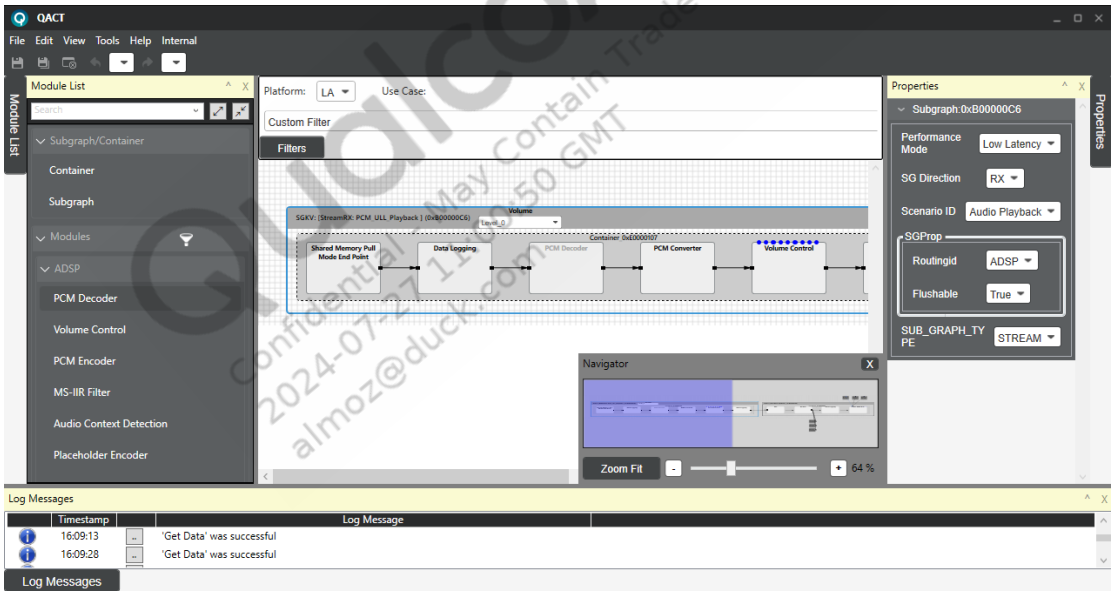
3.1.4 Use calibration files

QACT can interact with the following calibration file types:

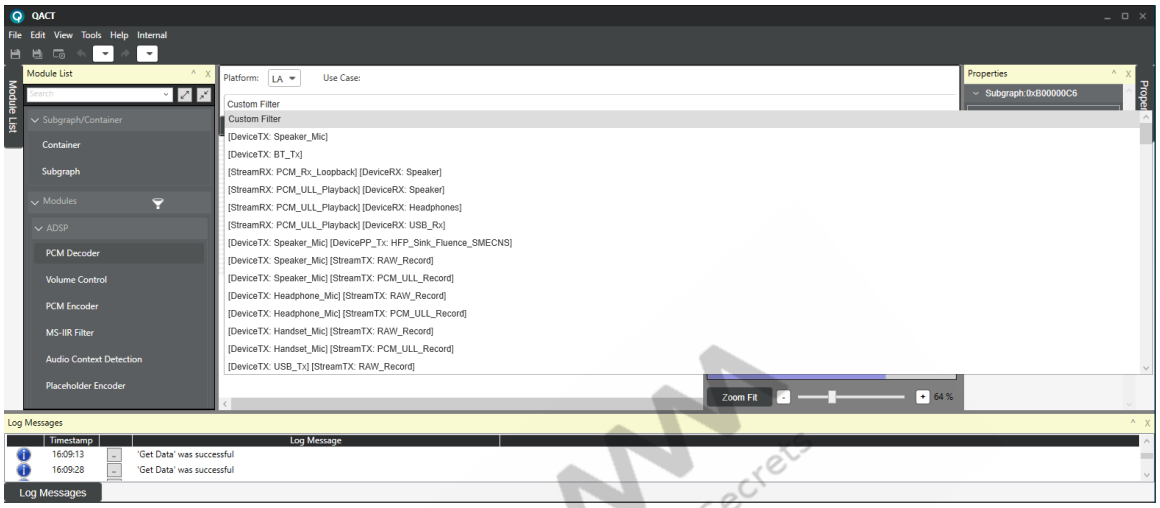
- .acdb file – The .acdb file is used to change existing parameters and tune values on the target without having to recompile the target's source files.
- Workspace file (.qwsp) – A workspace file is required when there is extra information that would not be supported in the *.acdb file. The workspace file is only used for QACT to store extra information needed by the UI. This may occur when new customized topology information is added. This information is stored in .qwsp files, not .acdb files.

To open a calibration file:

1. Click **Open File** from the home screen.
2. Select a workspace (*.qwsp) file and click **Open**.
3. Canvas View appears, displaying the available use cases and parameters for the calibration file.



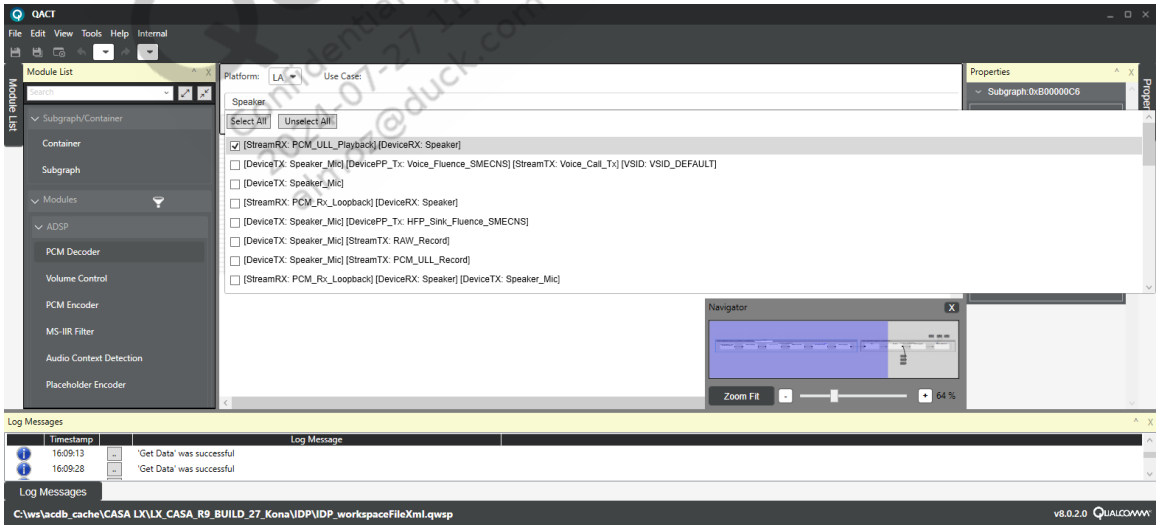
- To select a use case, click the **Use Case** drop-down and make a selection from the list.



- To create a custom filter of use cases, click **Filters**. Select a custom filter of the use cases to show multiple use cases at the same time.

  This can be achieved by using the top search bar to search for a specific term (for example, “Speaker") to reduce the use case list to only matching entries.

  To select multiple use cases, select the checkbox next to use cases of interest.



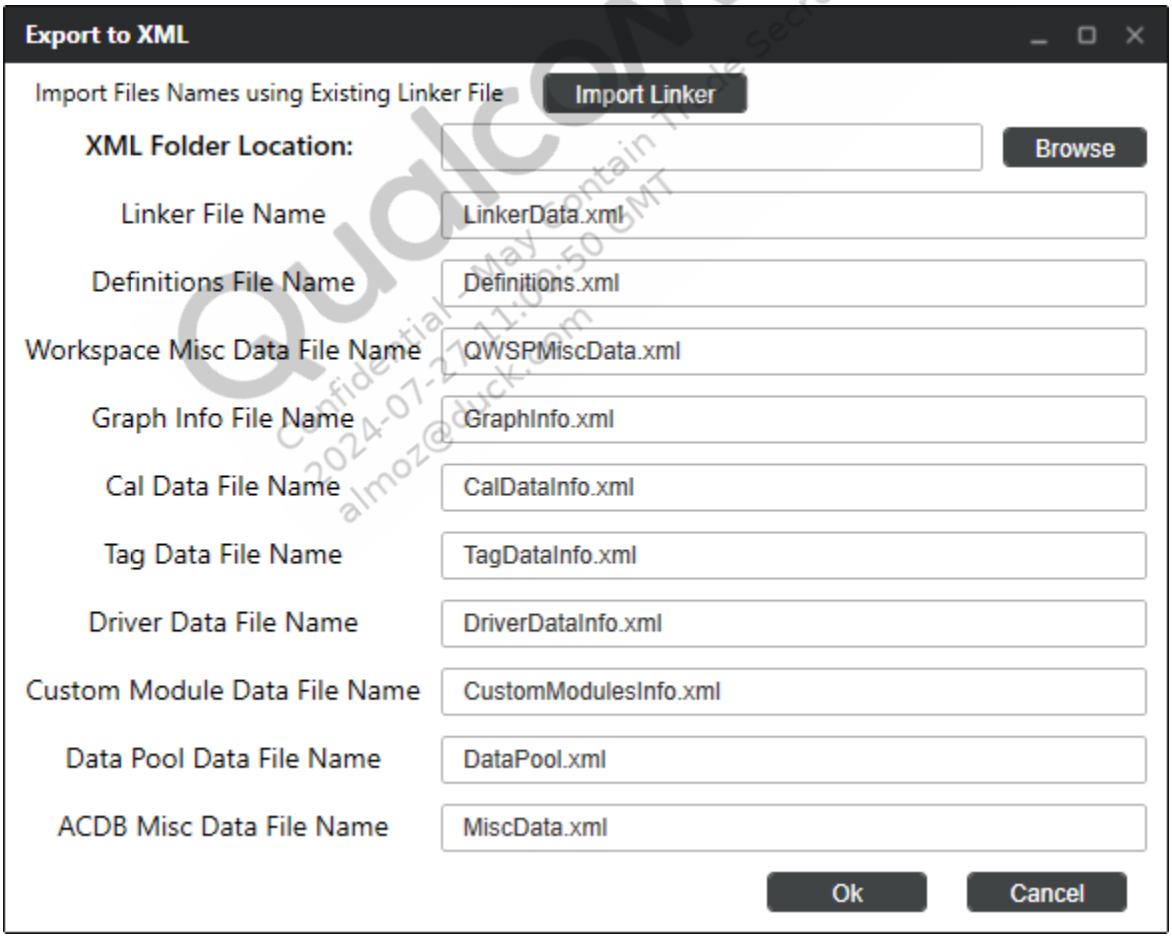
3.1.5 ACDB data in XML format

QACT supports saving ACDB/QWSP files to XML format. This provides opportunity to third party tools to use ACDB data

3.1.5.1 Export to XML format

To export QWSP/ACDB contents to XML format, click **File -> Export To XML**. ACDB/QWSP files content will be split to several XML files for easy understanding of the data and the organization of data. XML files and their contents are explained in detail in Section 3.1.5.3.

The following dialog allows selection of a folder where all generated XML files will be saved. Each XML file name can be changed. LinkerData.xml is the main file that links all generated XML files. XML file names can be imported using the existing LinkerData.xml file.



3.1.5.2 Open XML data

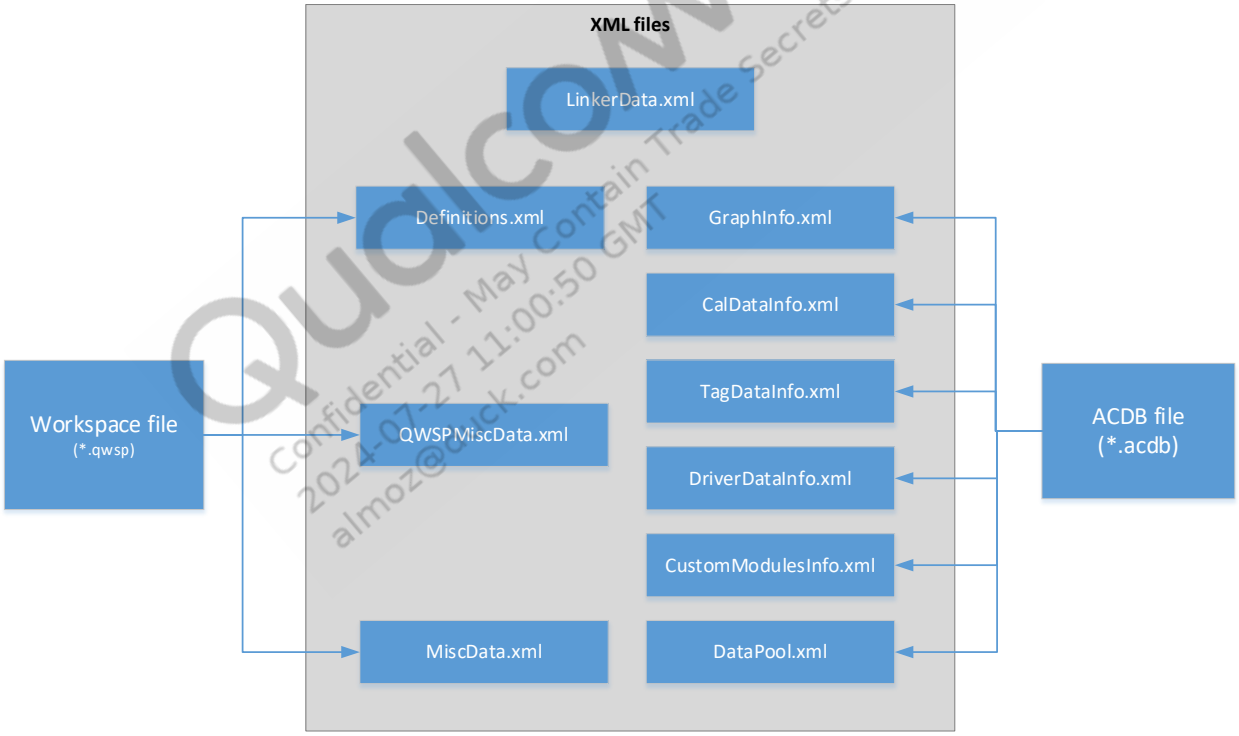
QACT also supports opening XML files and saving back to QWSP/ACDB format. To open an XML file, in the QACT launcher window, click **Open ACDB File on Disk** and select LinkerData.xml. All linked XML files will open and QACT will perform some validations and create a new session with the XML file data.

This data can be saved to QWSP/ACDB files using **File -> Save** or can be exported to XML format using **File -> Export to XML**.

3.1.5.3 XML data format

3.1.5.3.1 XML file list

Th following image explains the XML files generated and their contents.



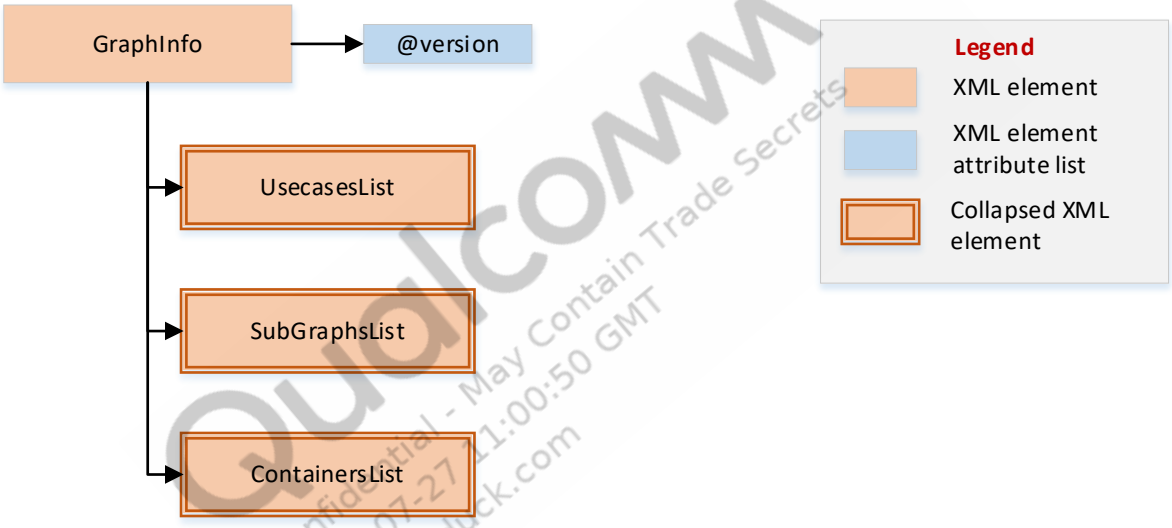
LinkerData.xml file links all generated XML files. It contains information for each XML file type and its name.

XML file name	File details
Definitions.xml	Contains all keys, modules, and property definitions. This data comes from the QWSP file.
QWSPMiscData.xml	Contains miscellaneous data from QWSP file such as UI persistent information
MiscData.xml	Contains miscellaneous data from QWSP and ACDB files such as ACDB file version information, header information, global properties, etc.
GraphInfo.xml	Contains the use case list, subgraph list, and container list

XML file name	File details
CalDataInfo.xml	Lists all calibration data instances for each subgraph/module/CKV combination
TagDataInfo.xml	Lists all tag data instances for each subgraph/module/TKV combination
DriverDataInfo.xml	Lists all driver data modules
CustomModulesInfo.xml	Contains custom modules information
DataPool.xml	Contains all PID payload data

3.1.5.3.2 GraphInfo.xml format

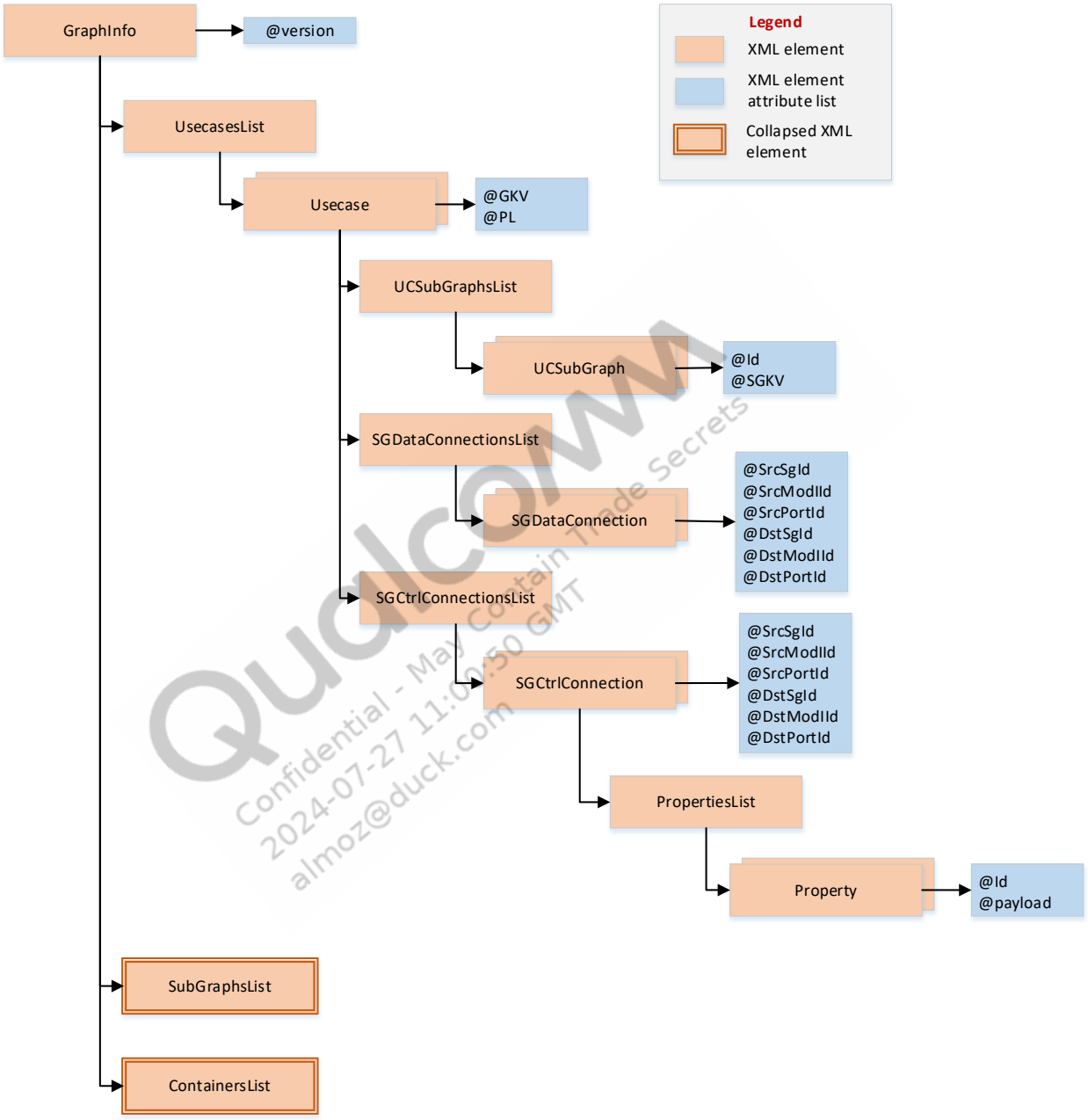
The following image shows the high level listing of elements.



The following table describes each XML element.

XML element name	Description
GraphInfo	Root element that holds all use cases, subgraphs, and the container list
UsecaseList	Contains the list of all use cases, their subgraphs and their data and control connection information
SubgraphList	Contains the list of all subgraphs, their modules, and their data and control connection information
ContainerList	Contains the list of all containers and their properties

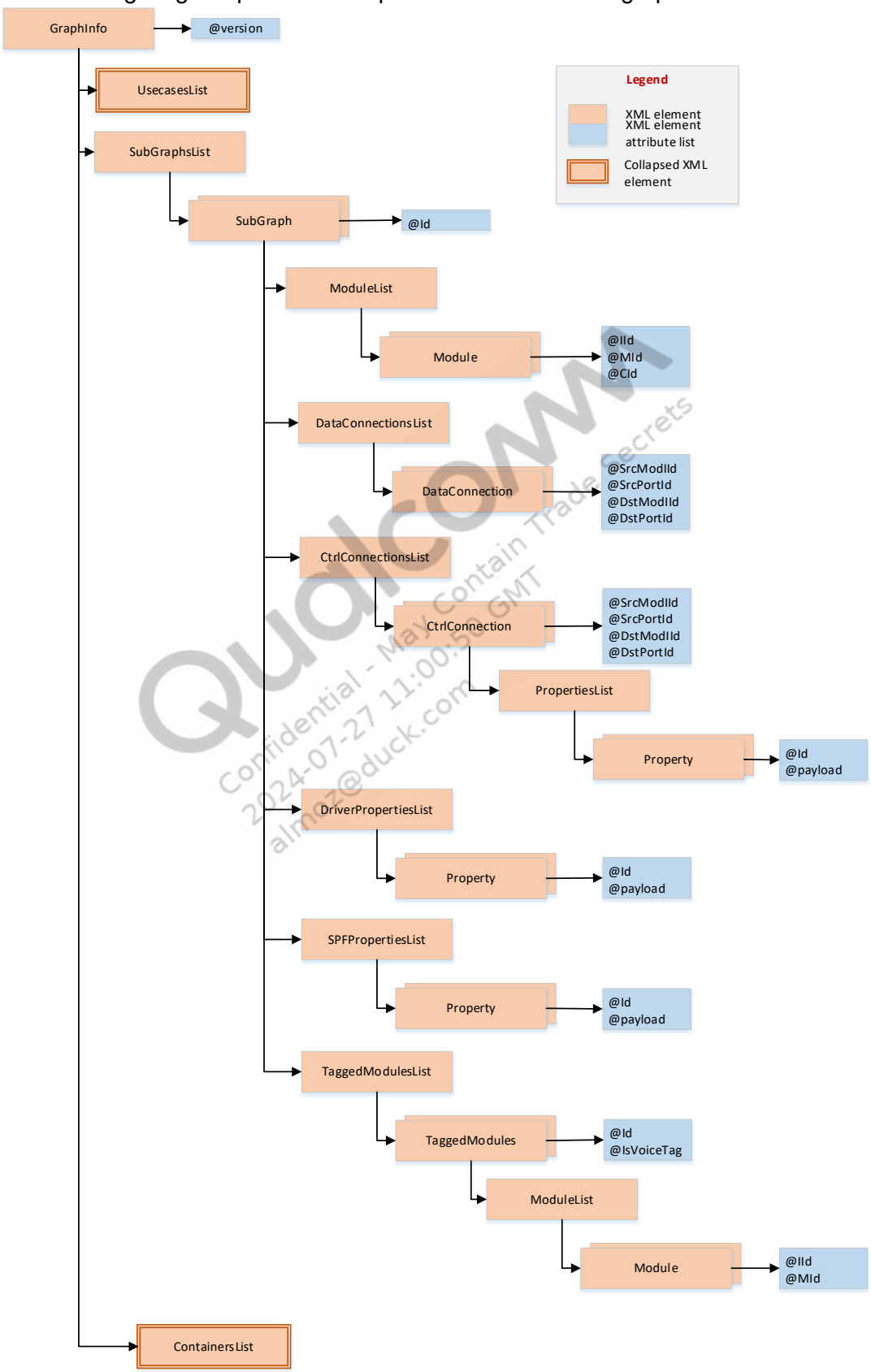
The following diagram provides complete details of the UsecaseList element.



The following table describes each XML element of UsecaseList.

XML element name	Description
UsecaseList	Contains the list of all use cases, their subgraphs, and their data and control connection information
Usecase	Each Usecase element holds information about one use case. It lists subgraphs and data/control connections between subgraphs. The following attributes are used to identify each use case: ▪ @GKV – A string format of graph key/value vector. Key/value is separated by a colon and each key/value pair is separated by a semicolon. For example, “key1:value1;Key2:value2”. ▪ @PL – Platform on which this use case runs. Useful only in the case of Hypervisor.
UCSubGraphrList	Contains the list of all subgraphs in the use case
UCSubGraph	Each UCSubGraph element holds information about one subgraph. The following attributes are used to identify each subgraph: ▪ @Id – Subgraph ID ▪ @SGKV – Subgraph GKV, a string format of the graph key/value vector. The key/value is separated by a colon and each key/value pair is separated by a semicolon. For example, “key1:value1;Key2:value2”.
SGDataConnectionsList	Contains the list of data connections between subgraphs
SGDataConnection	Holds the information of one data connection between the subgraphs of a use case. The following attributes are used to describe a data connection: ▪ @SrcSgId – Subgraph ID from where data connection originated ▪ @SrcModIId – Module instance ID from where data connection originated ▪ @SrcPortId – Port ID from where data connection originated ▪ @DstSgId – Subgraph ID where data connection ended ▪ @DstModIId – Module instance ID where data connection ended ▪ @DstPortId - Port ID where data connection ended
SGCtrlConnectionsList	Contains the list of control connections across subgraphs
SGCtrlConnection	Holds the information of one control connection between modules. The following attributes are used to describe a control connection: ▪ @SrcSgId –Subgraph ID from where control connection originated ▪ @SrcModIId – Module instance ID from where control connection originated ▪ @SrcPortId – Port ID from where control connection originated ▪ @DstSgId – Subgraph ID where control connection ended ▪ @DstModIId – Module instance ID where control connection ended ▪ @DstPortId - Port ID where control connection ended
PropertiesList	Contains the list of control connection properties
Property	Holds the payload of one connection property. ▪ @Id – Property ID ▪ @payload – Property payload in hexBinary format

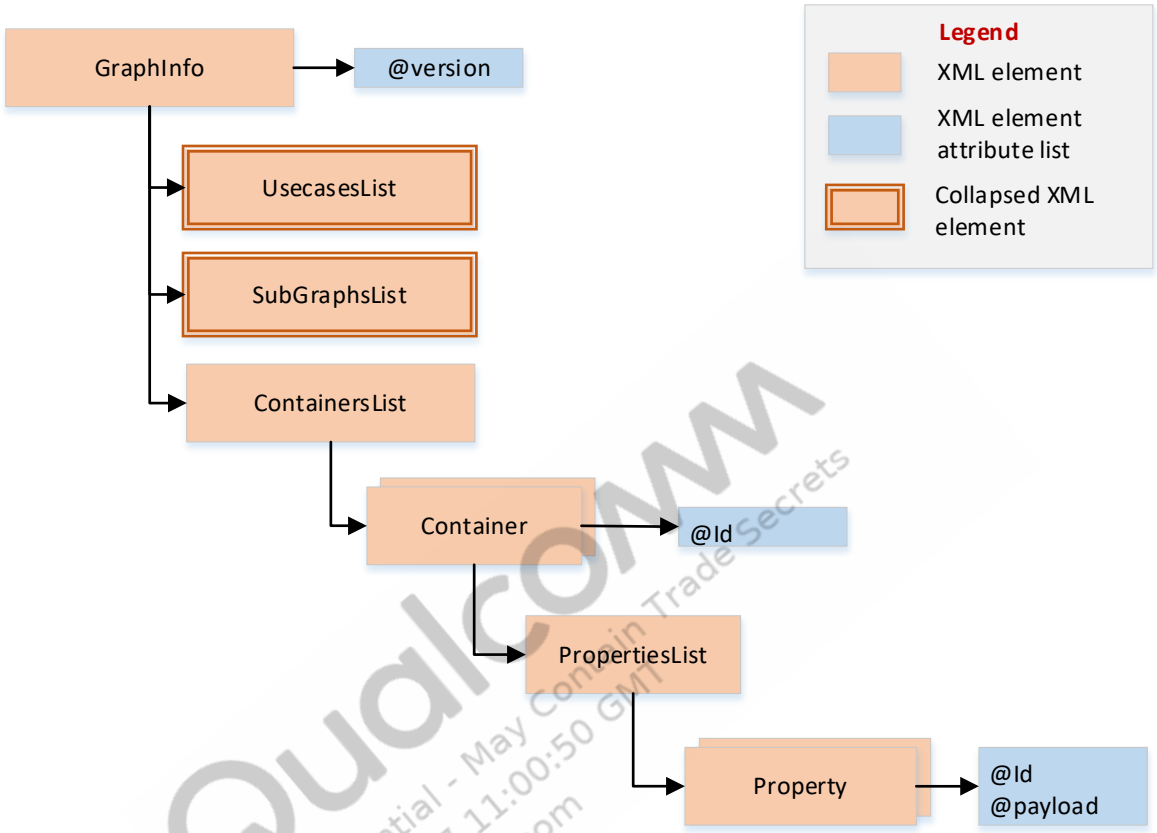
The following diagram provides complete details of the SubgraphsList element.



The following table describes each xml element of SubgraphsList.

XML element name	Description
SubGraphrList	Contains the list of all subgraphs and their module information in the use case
SubGraph	Each subgraph element holds information about one subgraph. The following attribute is used to identify each subgraph: ▪ @Id – Subgraph ID
ModuleList	Contains the list of modules in the subgraph
Module	Contains information about one module. The following attributes are used to identify each module: ▪ @IId – Module instance ID ▪ @Mid – Module ID ▪ @Cid – Container ID, identifies which container the module is part of
DataConnectionsList	Contains the list of data connections between modules
DataConnection	Holds information of one data connection between modules of a subgraph. The following attributes are used to describe a data connection: ▪ @SrcModIId – Module instance ID from where data connection originated ▪ @SrcPortId – Port ID from where data connection originated ▪ @DstModIId – Module instance ID where data connection ended ▪ @DstPortId - Port ID where data connection ended
CtrlConnectionsList	Contains the list of control connections between modules
CtrlConnection	Holds information of one control connection between modules. The following attributes are used to describe a control connection. ▪ @SrcModIId – Module instance ID from where control connection originated ▪ @SrcPortId – Port ID from where control connection originated ▪ @DstModIId – Module instance ID where control connection ended ▪ @DstPortId - Port ID where control connection ended
PropertiesList	Contains the list of control connection properties or a driver (GSL) property
Property	Holds the payload of one connection property or a driver (GSL) property. ▪ @Id – Property ID ▪ @payload – Property payload in hexBinary format
DriverPropertiesList	Contains the GSL properties list of the subgraph
SPFPropertiesList	Contains the SPF properties list of the subgraph
TaggedModulesList	Contains the tagged module list
TaggedModules	Contains the list of tagged modules. The following attributes are used to describe tagged modules: ▪ @Id – Tag ID ▪ @IsVoiceTag – Boolean value indicating if this is a voice tag

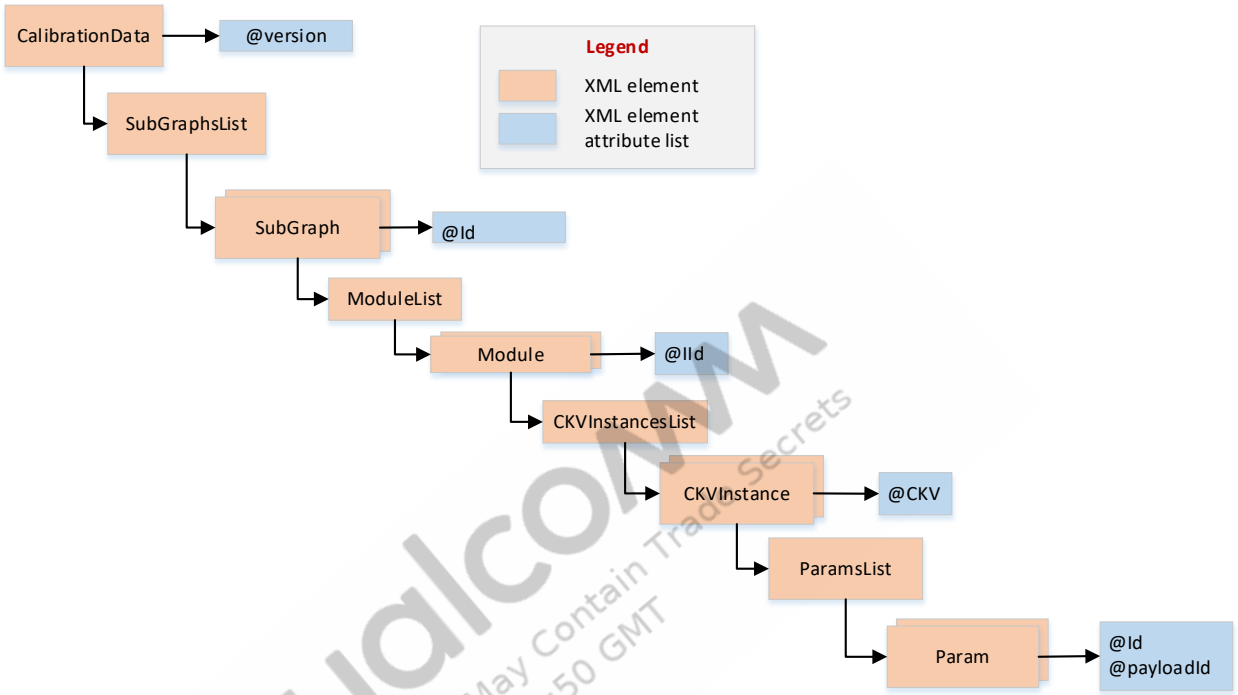
The following diagram provides complete details of ContainersList.



XML element name	Description
ContainersList	Contains the list of all containers information in the use case
Container	Each container element holds information about one container. The following attribute is used to identify each subgraph: ▪ @Id – Container ID
PropertiesList	Contains the list of container properties
Property	Holds the payload of one container property ▪ @Id – Property ID ▪ @payload – Property payload in hexBinary format

3.1.5.3.3 CaldataInfo.xml

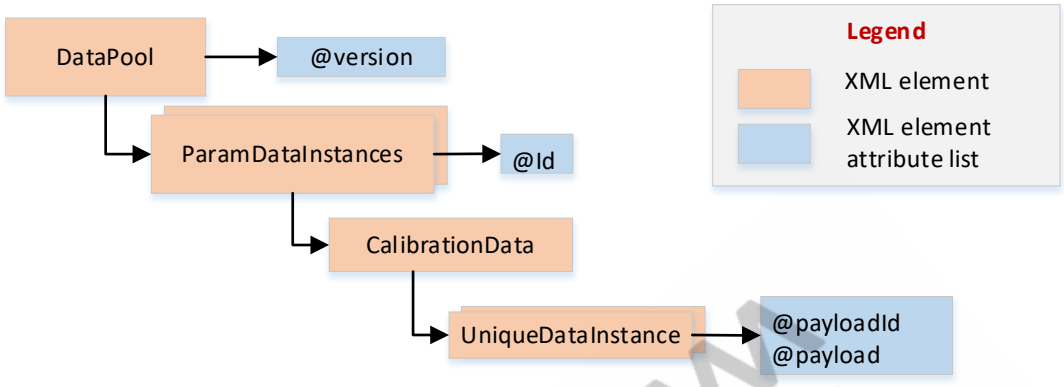
The following diagram is the layout of CaldataInfo.xml elements.



XML element name	Description
CalibrationData	Root element of CalibrationDataInfo.xml
SubGraphrList	Contains list of all subgraphs and their modules' calibration data
SubGraph	Each SubGraph element holds information about one subgraph and their modules' calibration data. The following attribute is used to identify each subgraph. ▪ @Id – Subgraph ID
ModuleList	Contains the list of modules in a subgraph
Module	Contains calibration data for all CKVs. The following attribute is used to identify each module: ▪ @IId – Module instance ID
CKVInstancesList	List of all CKVs and calibration data for the CKV
CKVInstance	Holds calibration data for one CKV. ▪ @CKV – Module CKV, a string format of calibration Key/value vector. Key/value is separated by a colon and each key/value pair is separated by a semicolon. For example, "key1:value1;key2:value2".
ParamsList	Contains list of PID of the module and their data
Param	Holds cal data for one PID. ▪ @Id - PID ▪ @payloadId – 64-bit unique ID (GUID format) that references to the payload in DataPool.xml.

3.1.5.3.4 DataPool.xml

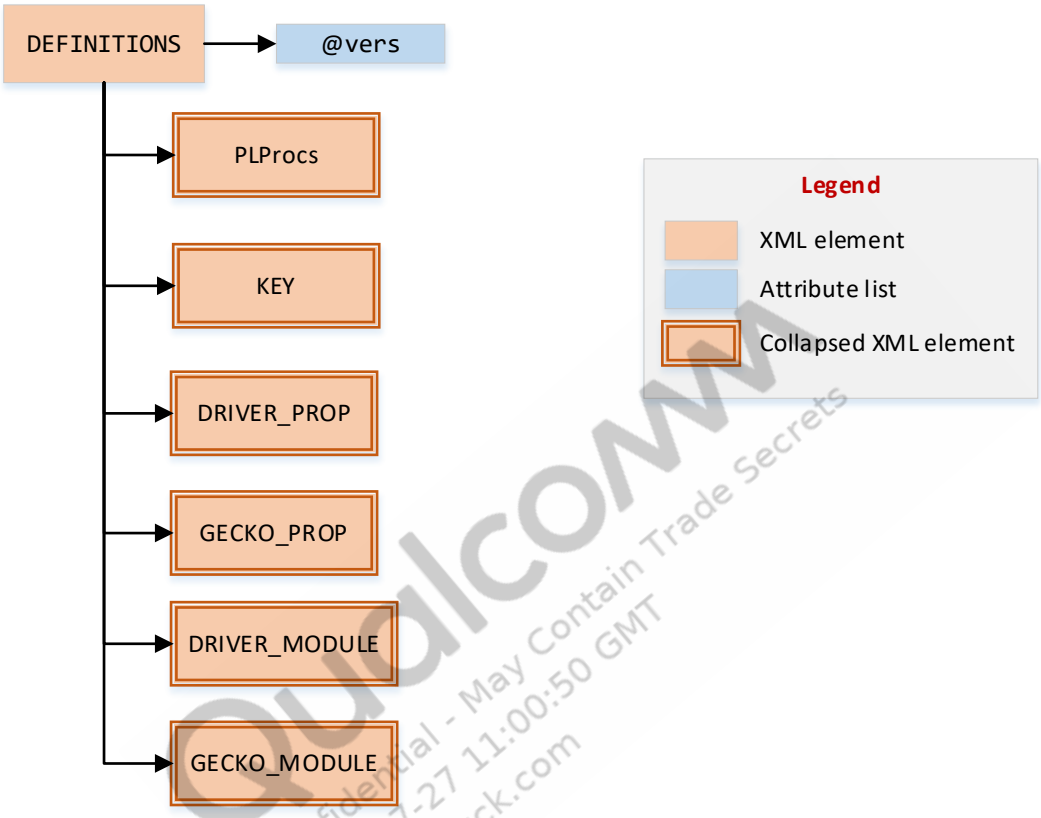
The following diagram is the layout of Datapool.xml elements



XML element name	Description
DataPool	Root element of Datapool.xml
ParamDataInstances	Contains all unique data instances of one PID ▪ @Id – PID
CalibrationData	List of unique data instances of a PID
UniqueDataInstance	Contains a unique cal data instance of a PID ▪ @payloadId – 64-bit unique ID (GUID format) used to reference the payload in CalDataInfo.xml ▪ @payload – PID payload int hexBinary format

3.1.5.3.5 Definitions.xml

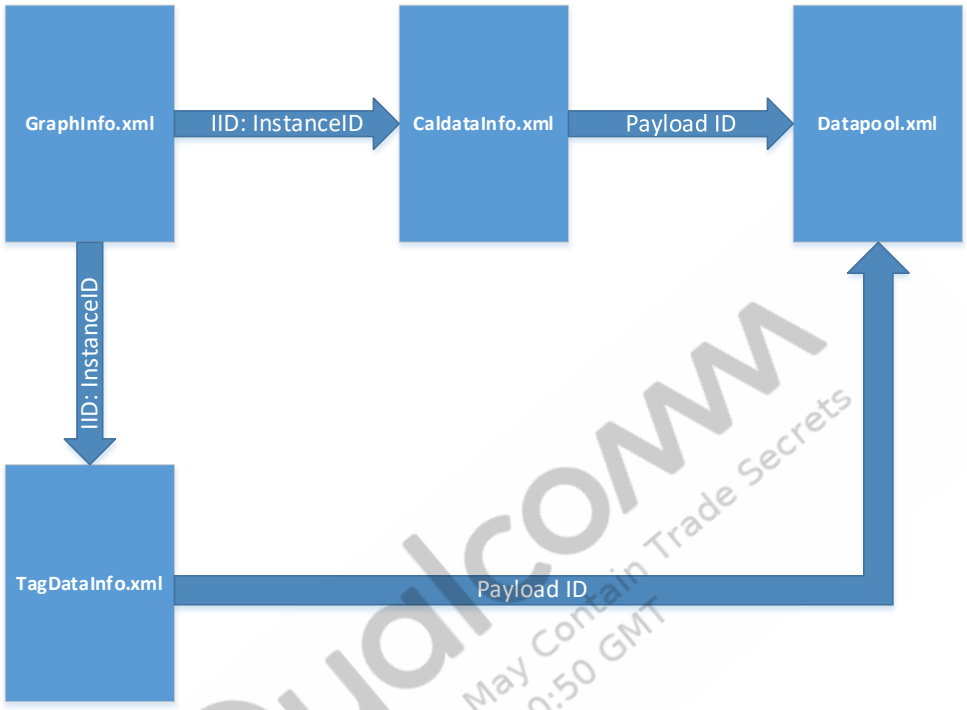
The following diagram shows the layout of the definitions XML file.



XML element name	Description
PLProcs	Contains the list of processors and their definitions
KEY	Contains definitions for all keys (GKVs, CKVs, etc.)
DRIVER_PROP	Contains definitions of all driver properties
GECKO_PROP	Contains definitions of all SPF properties
DRIVER_MODULE	Contains all driver modules and their definitions
GECKO_MODULE	Contains all SPF modules and their definitions

3.1.5.4 Understanding calibration data

The following diagram shows how each file is connected.



3.1.5.4.1 How to find module data when GKV and CKV is known

1. Find the module instance ID (IID) from GraphInfo.xml:
   a. Iterate UseCase in UseCaseList, find the GKV of interest, and obtain the SGList (list of UCSubGraph).
   b. Iterate SubGraph in SubGraphsList and find the subgraph of interest.
   c. For the matched subgraphs and iterate ModulesList to find the IID of the specific MID you are interested in.
2. Find payloadID from CalDataInfo.xml:
   a. For the subgraph of interest, iterate ModulesList and find the IID of the specific MID you are interested in.
   b. For the IID of interest, iterate CKVInstancesList to find the CKVs of interest.
   c. For each CKV and PID, make note of the payloadId; this is needed to lookup the calibration data in the datapool.xml.
3. For a given payloadId, look up the calibration data payload:
   a. Iterate ParamDataInstances to find the PID of interest.
   b. Iterate CalibrationData to find the UniqueDataInstance with a matching payloadId of interest.
   c. The PARAMETER_DATA element contains the PID payload in name/value pair format.

3.2 Debug the WCD/WSA hardware interface

3.2.1 Open the ADIE Register View

NOTE: For LA targets, the following commands must be run before using ADIE:

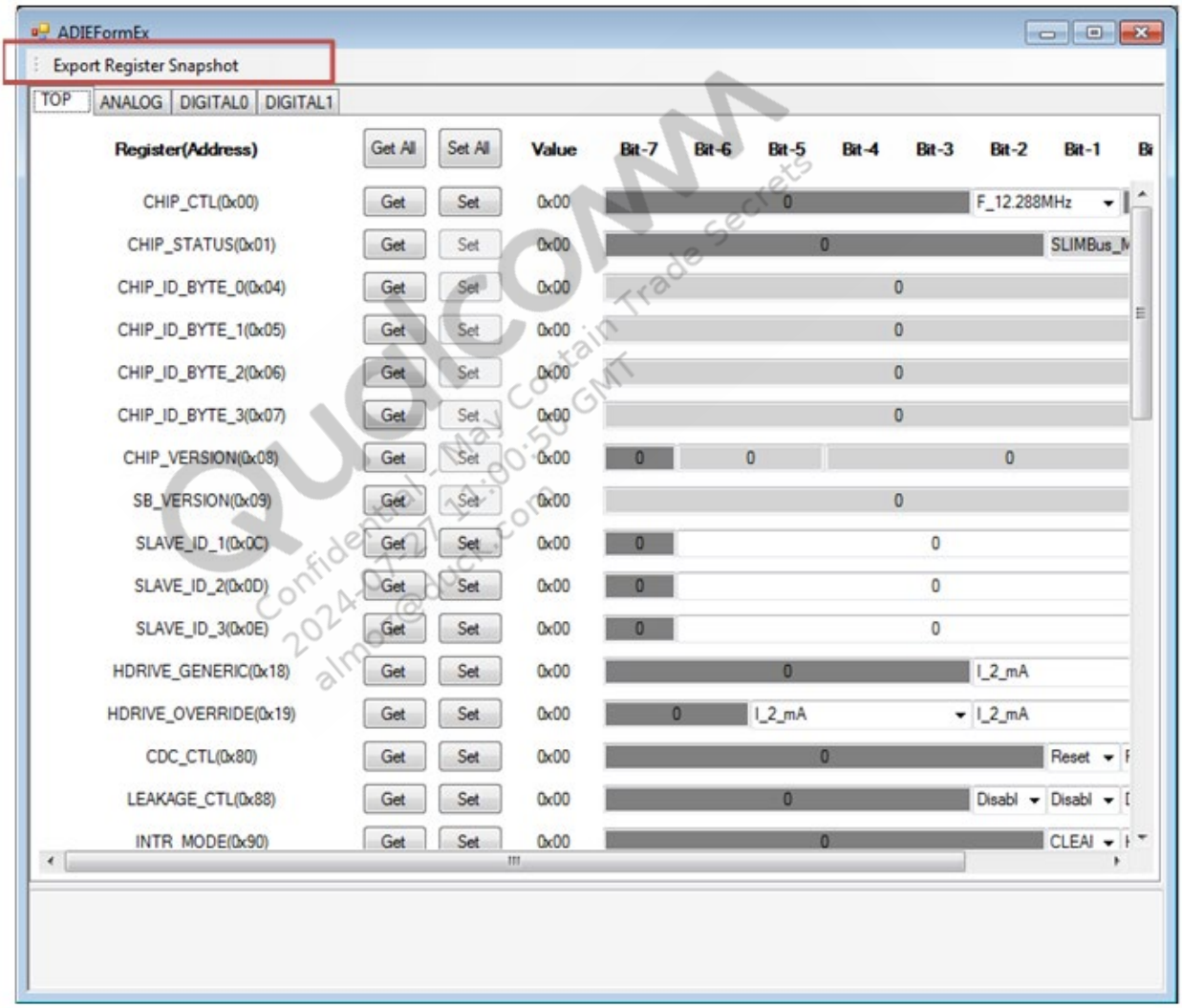
```
adb root
adb shell
adb remount
chmod 777 /sys/kernel/debug/asoc/*-snd-card/*_codec/codec_reg
```

1. Click **ADIE RTC** on the QACT home screen.
2. Click **Refresh** to access the content of all ADIE registers from the target device and load them in the ADIECalibratorTable.
3. Click the boxes in Bit 0 to Bit 7 to configure a value in a register. A check should appear in boxes representing bits that contain 1.
4. Click **Commit** to load values entered in the ADIECalibratorTable to the target device.

3.2.2 Export a register snapshot

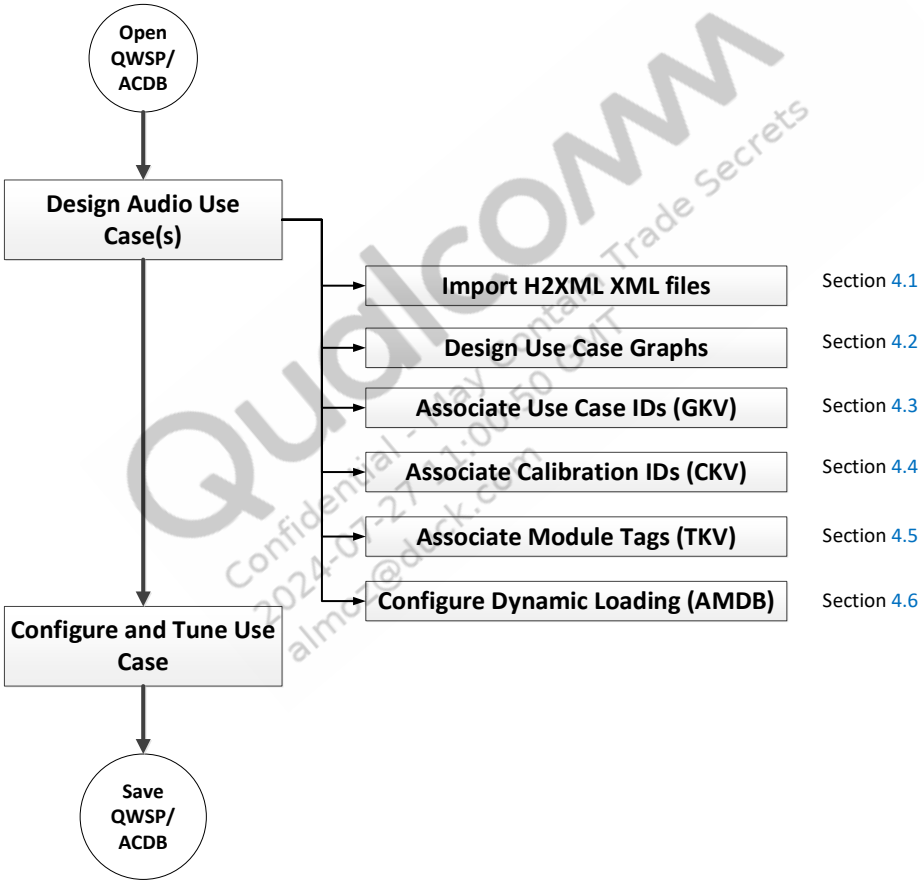
This feature allows users to dump a register snapshot into a readable ASCII format. This helps to analyze the ADIE register settings offline. The exported file can be opened in QCET for analysis.

1. Click **ADIE RTC** on the QACT home screen .
2. Select **Menu** > **Export Register Snapshot**.
3. Enter a filename in the Save dialog box and click **OK**.



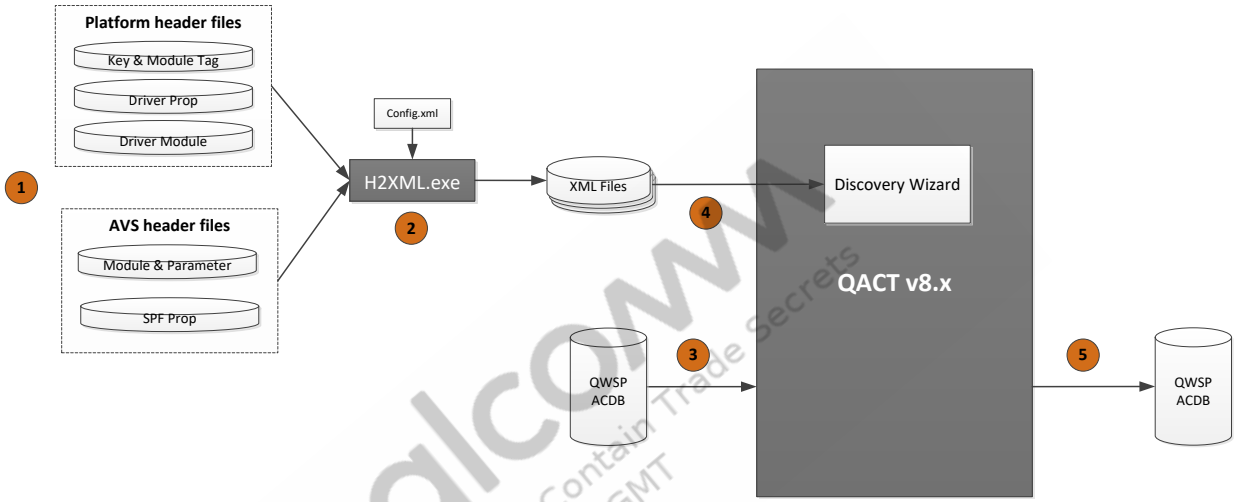
4 System designer

This role is responsible for designing audio use cases to meet product, software, and firmware requirements. The following diagram shows the system designer workflow in QACT.



4.1 Import H2XML files

H2XML is a tool to convert C header files with predefined H2XML annotations to XML files. These header files and converted XML files contain data structures (definitions), which are imported into QACT through DefinitionDiscoveryWizard. QACT supports these definitions and the System Designer can use these definitions to create use cases. The H2XML work flow is shown in the following diagram.



In this diagram:

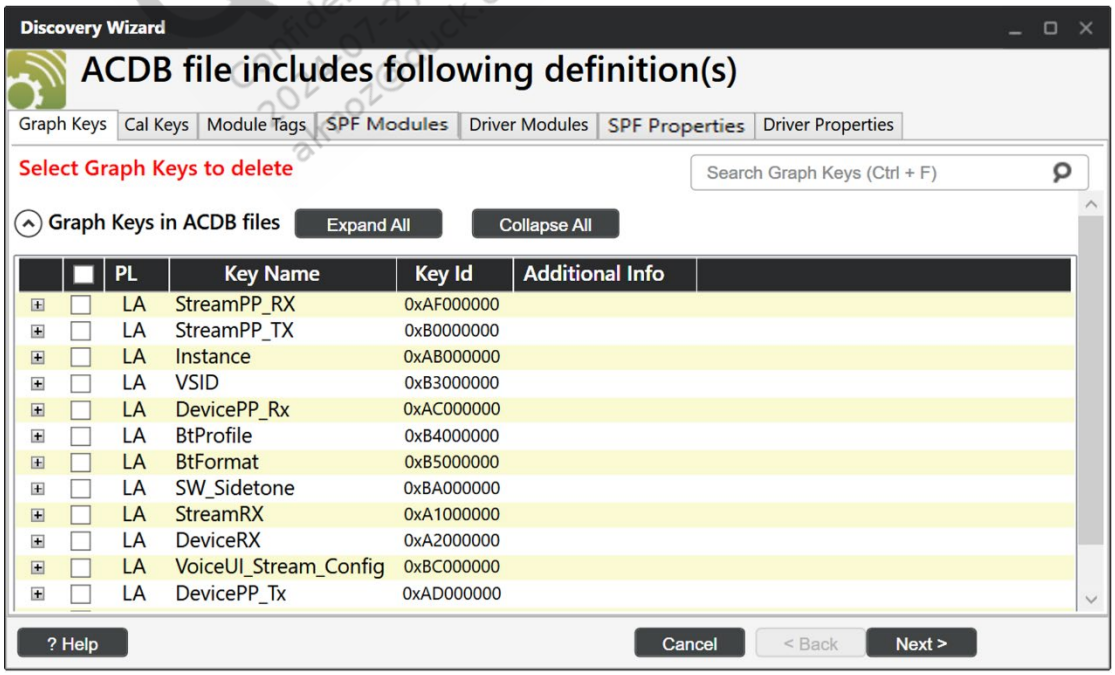
1. Platform and/or AVS software engineers add/edit header files for definitions (modules, parameters, key and module tags, driver properties, SPF properties) by adding H2XML annotations.
2. The H2XML.exe is used to convert header files to XML files.
3. QWSP and ACDB files are opened in QACT.
4. Converted XML files are imported to QACT through **Tools -> Discovery Wizard**.
5. Data is saved to QWSP and ACDB files.

The subsequent sections describe steps 4 and 5 in greater detail.

4.1.1 View definitions in ACDB

1. After opening a QWSP file, click **Tools -> Discovery Wizard**.
2. Click **View Defs in ACDB** to show all definitions in the currently selected ACDB file.

Each tab in Discovery Wizards shows one type of definition.



3. . Click one of the tabs described in the following table to view the corresponding definitions.

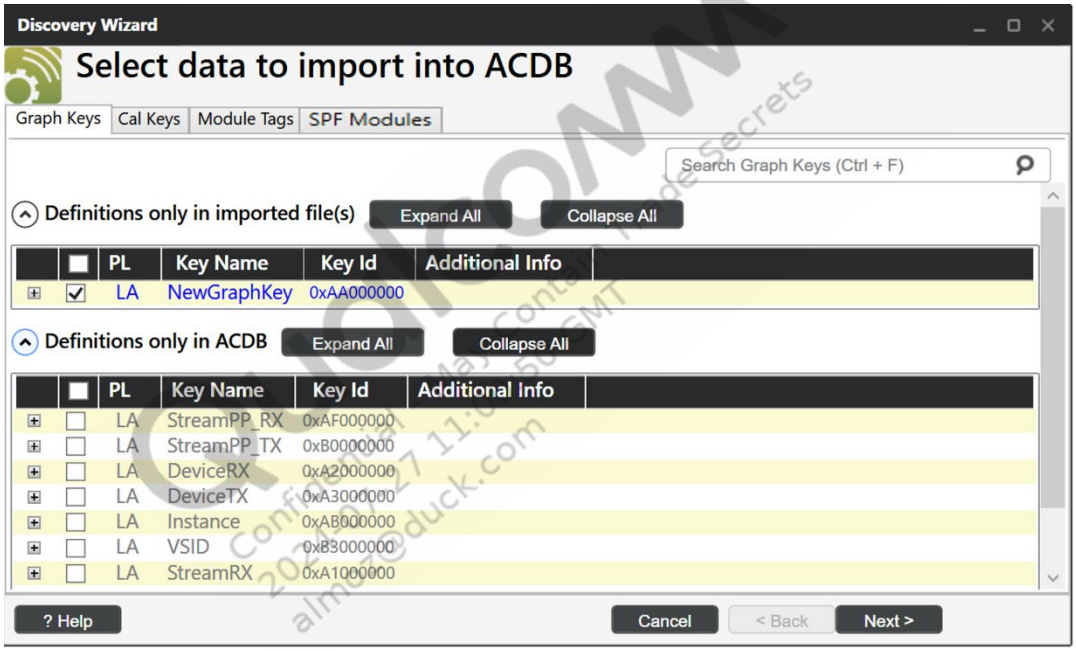
Tab name	Description	Defined in
Graph Keys	Key definitions for graphs to configure GKV	Platform software
Cal Keys	Key definitions for module calibration data (CKV)	Platform software
Module Tags	Module tag definitions for tagging modules and TKV	Platform software
Driver Modules	Driver module definitions; can be accessed from Tools -> Driver Module	Platform software
Driver Properties	Driver property definitions for configurating subgraphs	Platform software
SPF Modules	SPF module definitions	AVS software
SPF Properties	SPF property definitions for configurating subgraphs and containers	AVS software

4. At this point, definitions can optionally be removed from the ACDB file.
   - To remove definitions, select them using the corresponding checkboxes and click **Next**. The definition and its related data will be deleted. For example, if an SPF module is selected and deleted from this window, it will be removed from all subgraphs using this module. All its calibration data will be removed from ACDB as well.
   - To proceed without removing definitions, click **Next**.

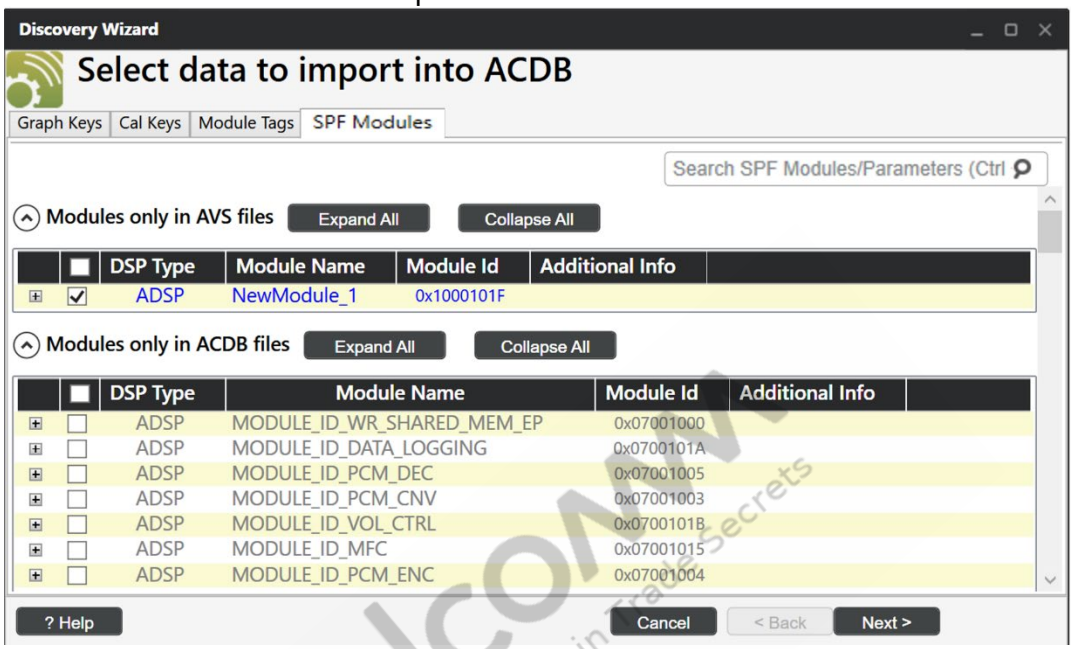
4.1.2 Add a definition to ACDB

To add a new module to an ACDB file:

1. Click **Tools -> Discovery Wizard**.
2. Click **Add**.
3. Select h2xml file(s) and click **Open**. Multiple h2xml files can be selected by holding Ctrl.
4. Click **Next**.
5. New definitions (which are not currently in the open ACDB file) are listed in the Definitions only in Imported file(s) section in blue. Select the definition to add using the checkboxes next to the definition names.



6. If more than one type of definition file has been imported, click the corresponding tabs to view and select other imported definitions.



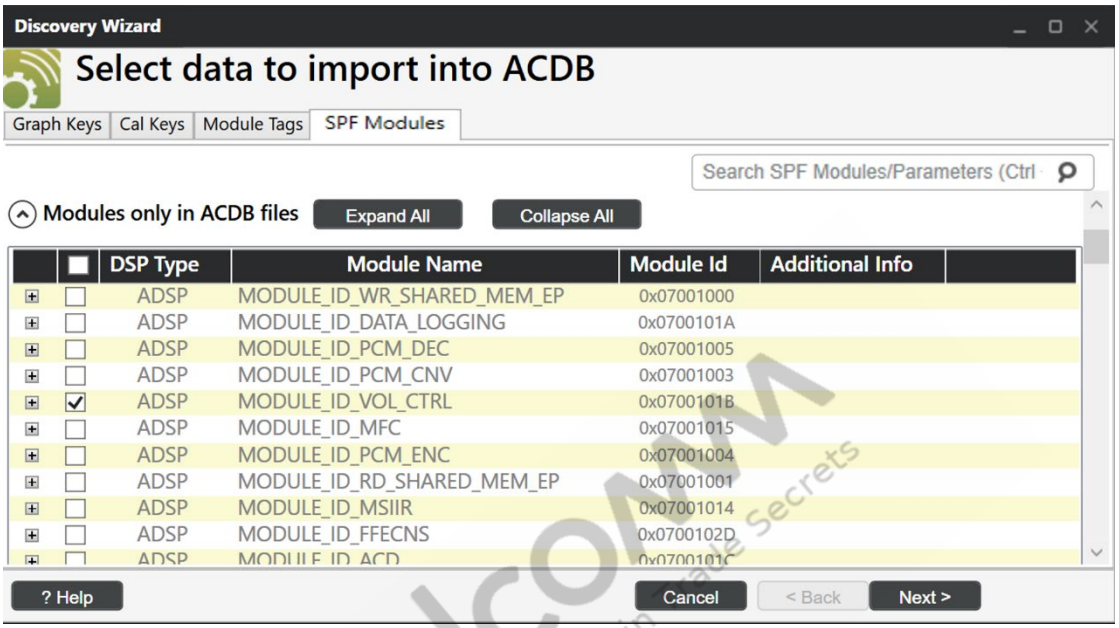
7. Click **Next**.
8. The definition(s) are added to the session and a summary of the updates is shown. Click **Finish** to close the wizard.

4.1.3 Delete a definition from ACDB

Definition can be deleted as described in 4.1.1 when viewing definitions in the ACDB file. Definitions can also be deleted while importing definition(s) from an h2xml file as follows.

1. Click **Tools -> Discovery Wizard**.
2. Click **Add**.
3. Select a h2xml file and click **Open**. Multiple h2xml files can be selected by holding **Ctrl**.
4. Click **Next**.

5. Definitions that can be deleted are listed under “Modules only in ACDB files”. Select the modules to delete using the checkboxes beside the module names.



6. Click **Next**.
7. The definition(s) are deleted from existing use cases, and definitions and a summary of the updates is shown. Click **Finish** to close the wizard.

4.1.4 Modify a definition in ACDB

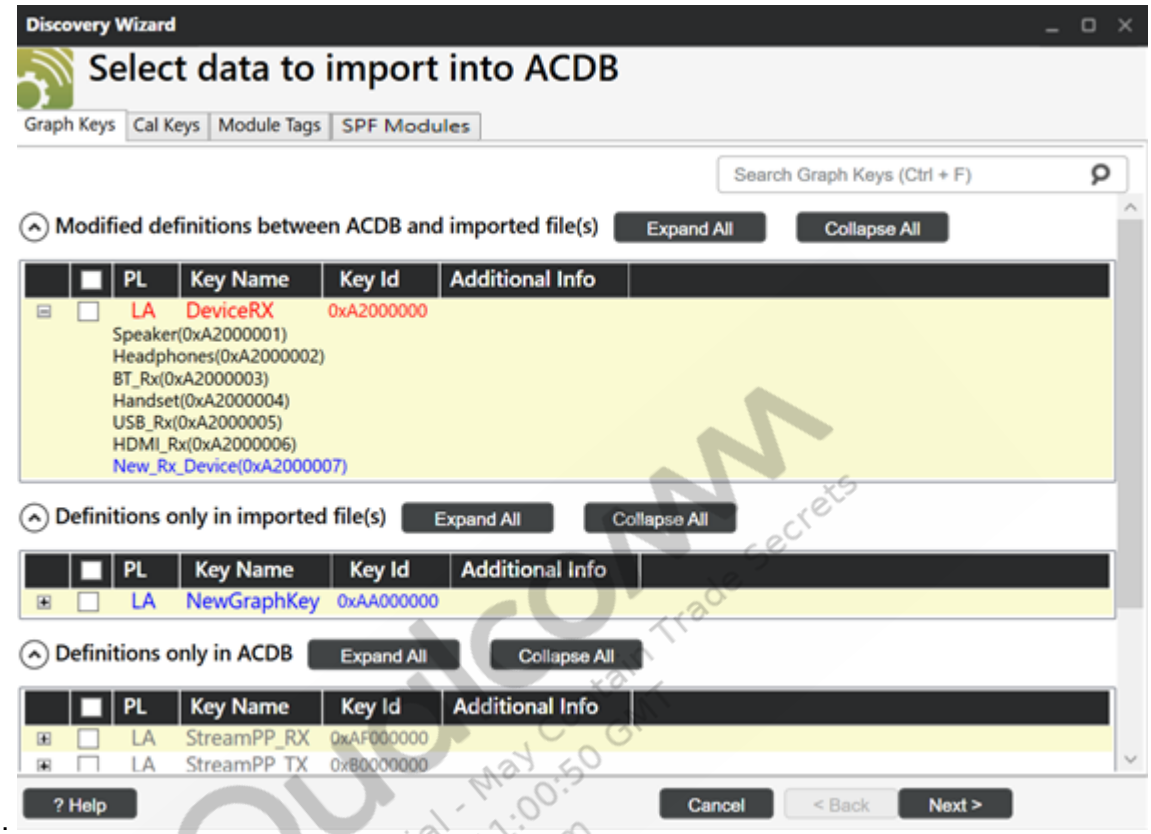
Modification includes the following changes to one or more contents within a definition (keys, modules, properties):

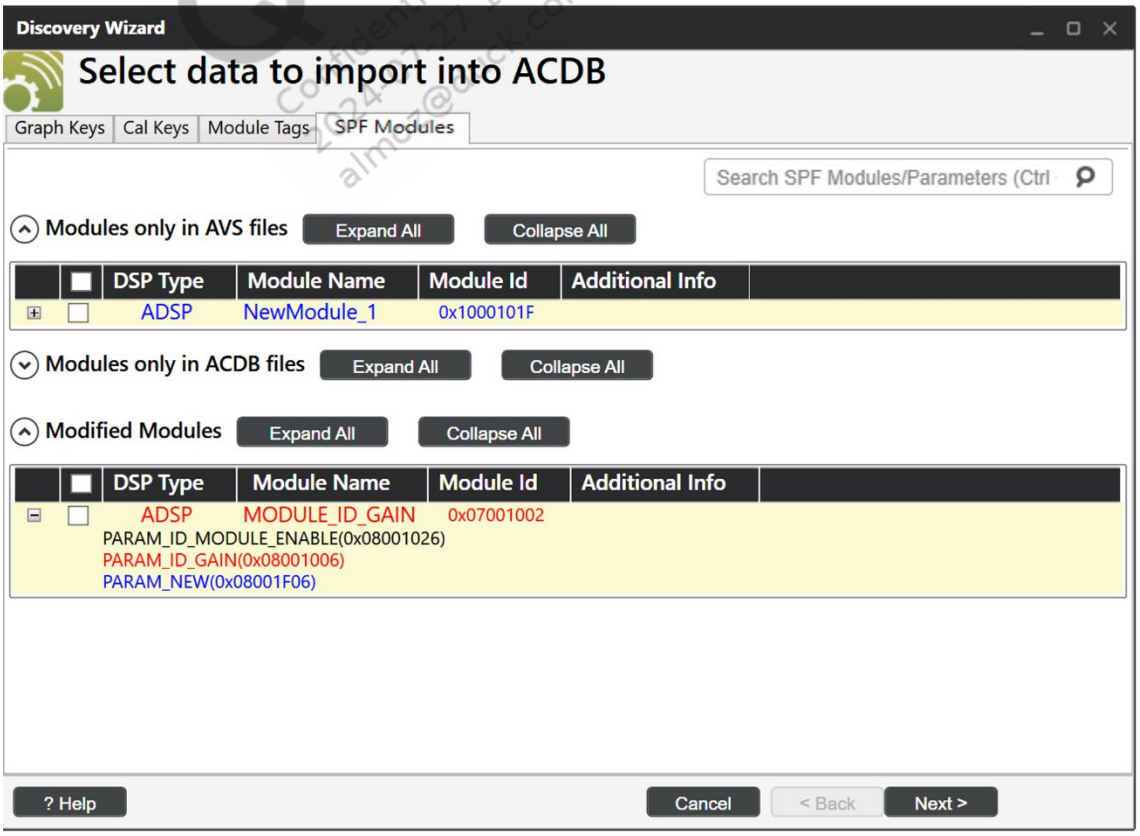
- Add/delete an item (key value, a module’s parameters, a property)
- Change the name or description
- Change an item’s default value, range, type

To modify a module:

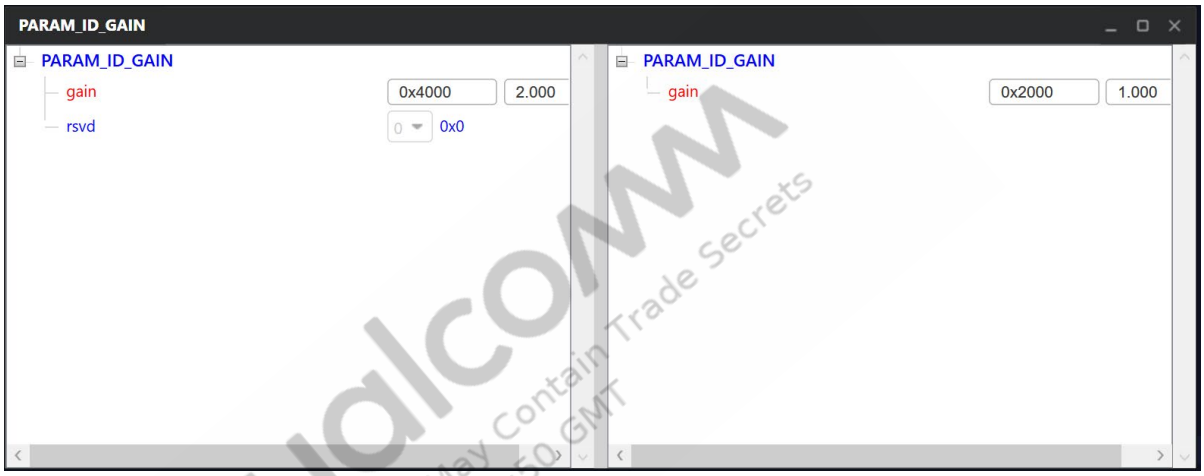
1. Click **Tools -> Discovery Wizard**.
2. Click **Add**.
3. Select a definition file(s) and click **Open**. Multiple definition files can be selected by holding **Ctrl**.

4. Click **Next.**





5. Definitions with available modifications are listed under Modified Definitions. The example on the previous page shows a modified Graph Key and SPF module. The modified graph key “DeviceRX” has a new value “New_Rx_Device” added. While the modified module “MODLE_ID_GAIN” has a modified parameter “PARAM_ID_GAIN” and a new added parameter “PARAM_NEW”.
   - Click **+** to see the parameters to be modified
   - Double click on an updated parameter to observe the differences in the UI. For example, PAEAM_ID_GAIN has a new item “rsvd” added, and the existing item “gain” default value is changed to 0x4000 from 0x2000.

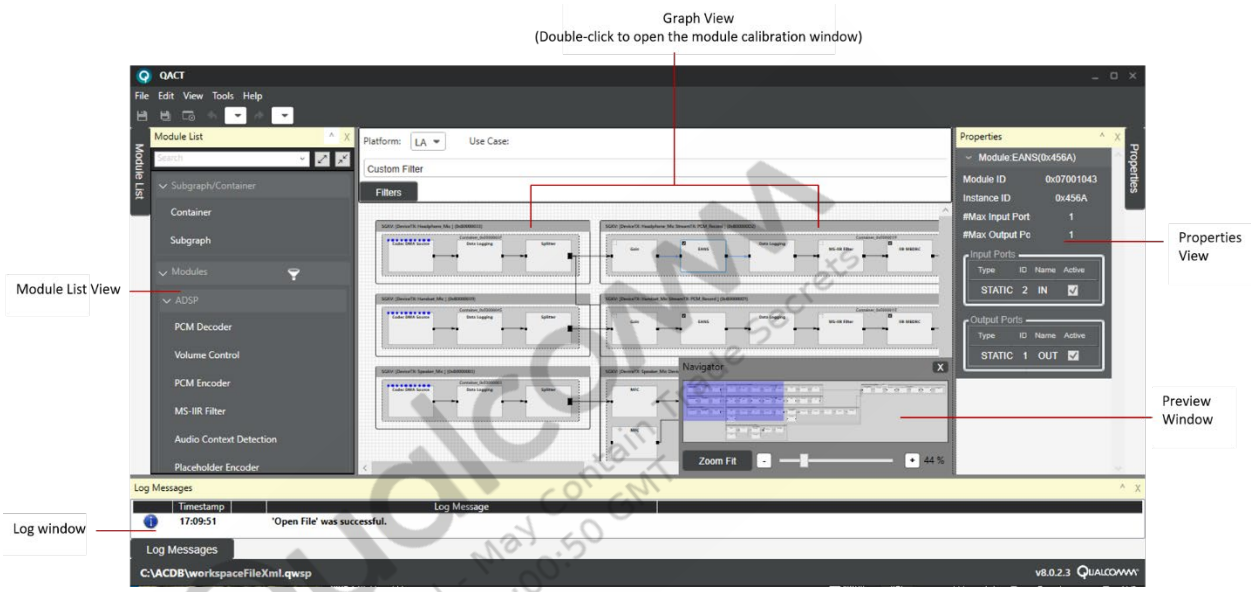


6. Select the definition to modify using the checkboxes beside the definition names.
7. Click **Next**.
8. The definition(s) are updated in all use cases, and definitions and a summary of the updates is shown. Click **Finish** to close the wizard.

4.2 Design use case graphs

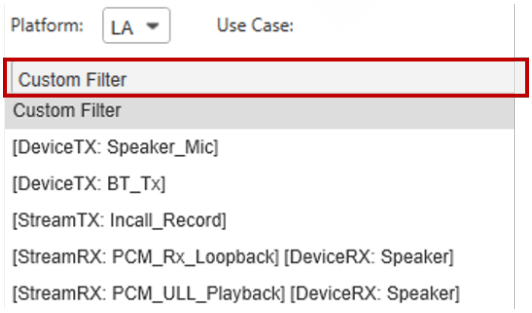
4.2.1 Introduction to QACT View

QACT View provides the capability to design graphs as well as tune modules. The components of the QACT UI are shown in the following image.

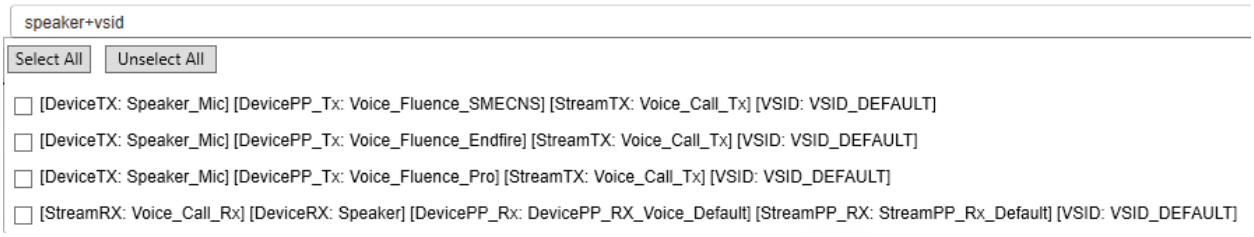


4.2.1.1 Graph View

Graph View displays audio modules in the order in which the graph is rendered in the DSP. Users can filter the GKVs by selecting the GKV from the dropdown menu or by using the Filter button.

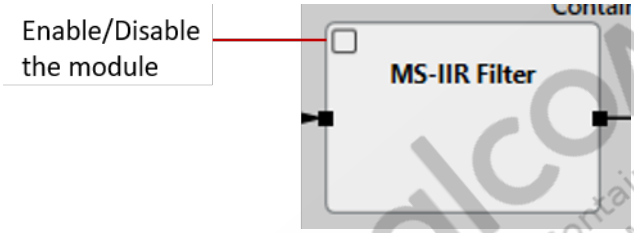


Click **Filters** to filter GKVs by typing keywords and selecting GKV(s). Then user needs to select one or more GKVs to display the selections.

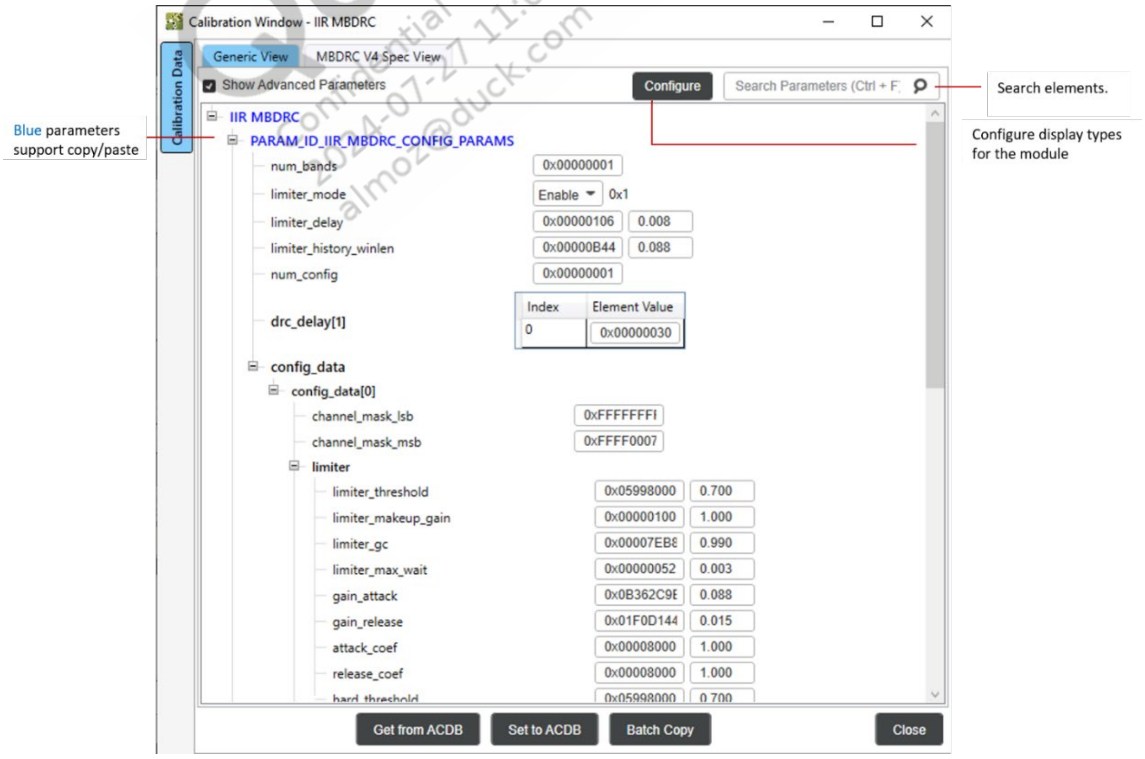


Once GKVs are selected, the user can modify the graphs or tune the modules.

Tuning modules can be enabled or disabled in graph view. When applicable, calibration values can be selected from the list displayed on the module.



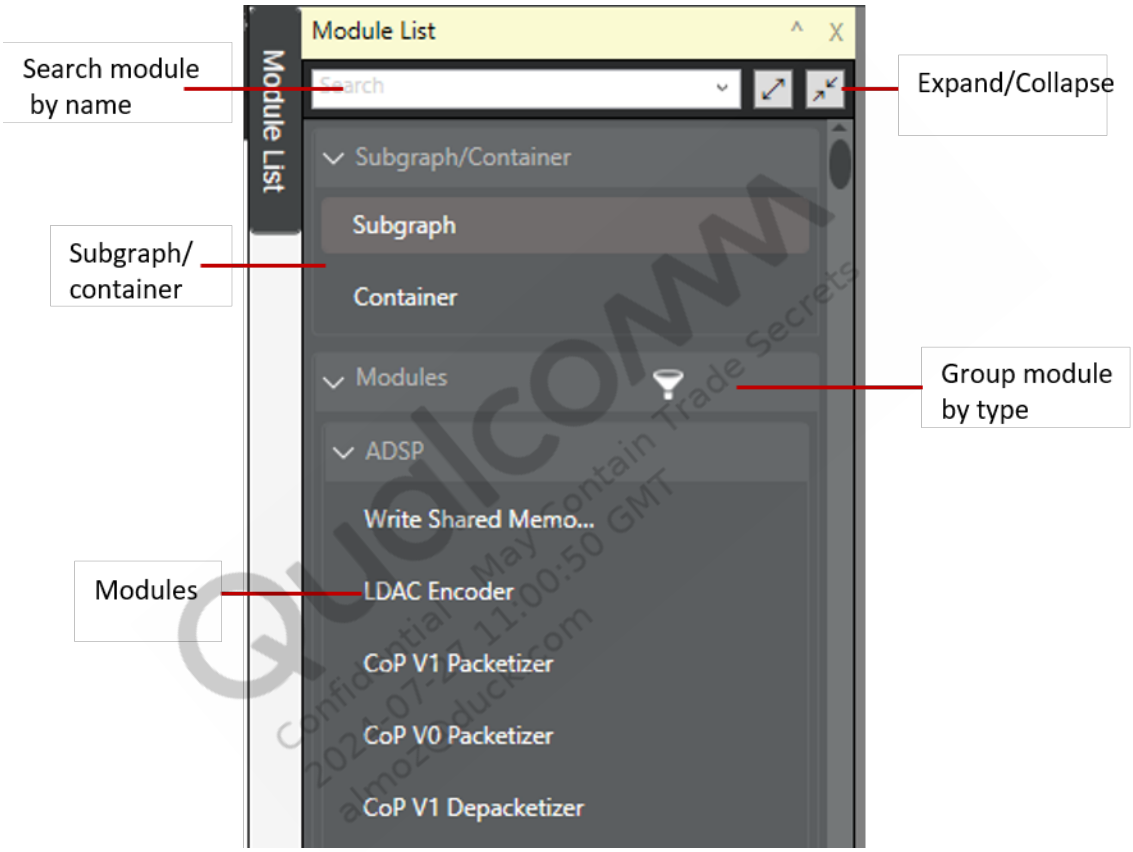
Double clicking a module in graph view opens the module calibration dialog.



4.2.1.2 Module List View

Module List View provides a list of all modules supported by the ACDB files in use. Users can drag and drop subgraphs, containers, and modules from this view to the Graph View to create graphs.

Module List View also provides the capability to search the list by module name or group modules by type.

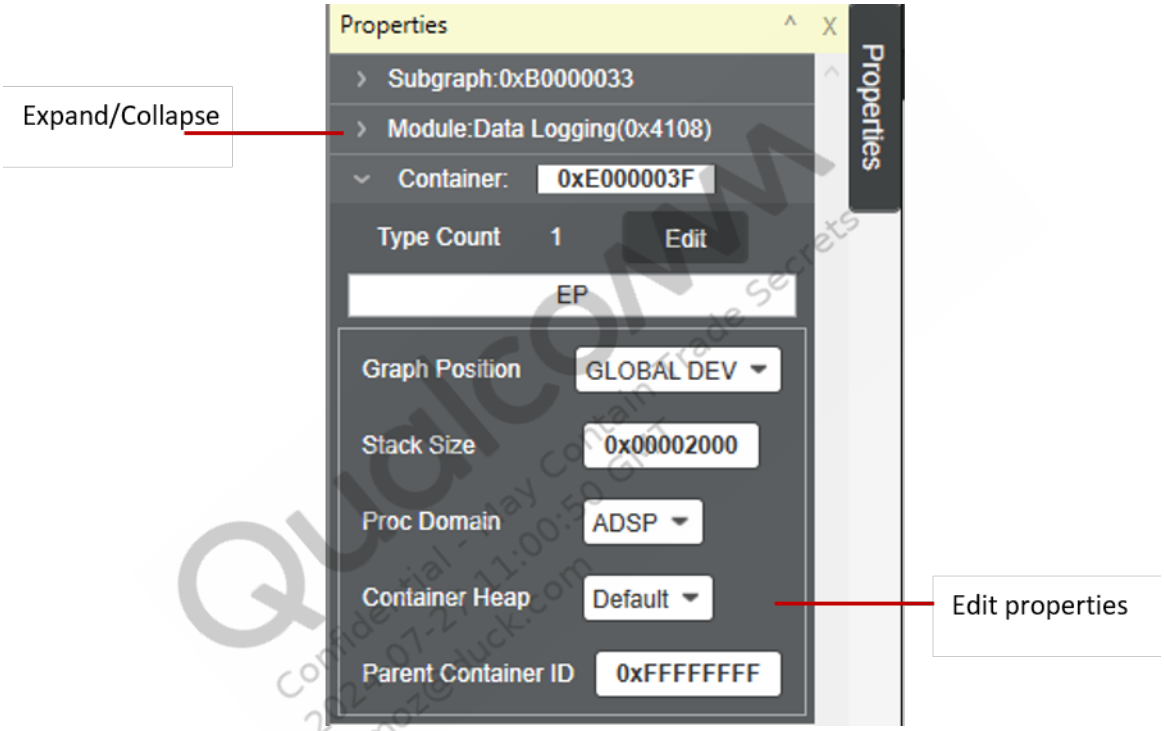


4.2.1.3 Properties View

Users can view and edit subgraphs, containers, and module properties using Properties View. Modifications made to the properties are automatically set after editing the value.

To view property of a subgraph, container, or module, click on the required element in Graph View. Users can also view multiple properties at a time by pressing **Ctrl** and selecting the required elements.

To clear Properties View, click on an empty region in Graph View.

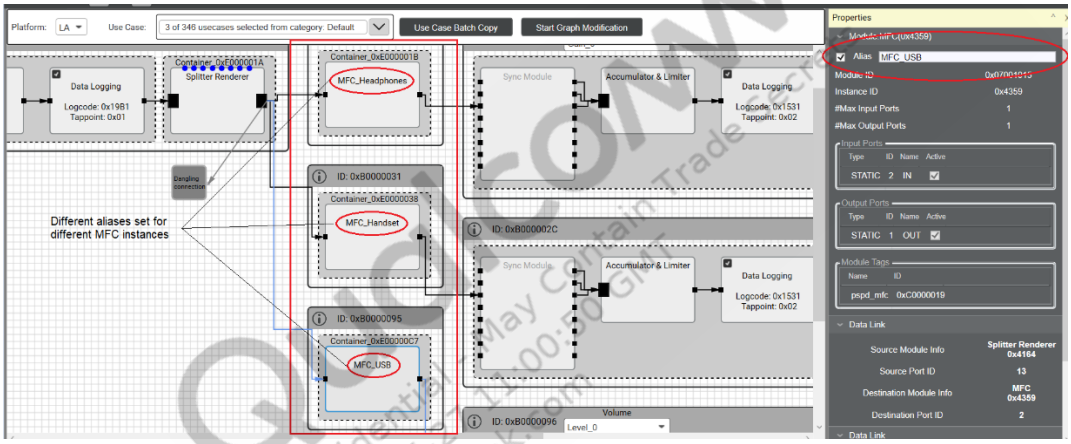


4.2.1.4 Module alias

Users can set an alias to each module instance for easy identification. This is particularly useful when there are multiple instances of the same module in the graph(s) shown in the graph view.

Module alias can be set in properties view, which shows up when a module instance is clicked by checking the check box and giving it a name. Once the alias is set, the information persists across sessions, so whenever the ACDB file is saved and the file is reopened, the alias that was set shows up for the module.

If an alias is set for a module, it will be shown in the graph designer view, properties view, module or use case batch copy, etc. so that it can be identified. Module aliases can also be merged between two ACDB files using the diff/merge feature and selecting the merge type as “Aliases and Metadata”.



4.2.2 Switch

A switch is an UI component in QACT graph view.

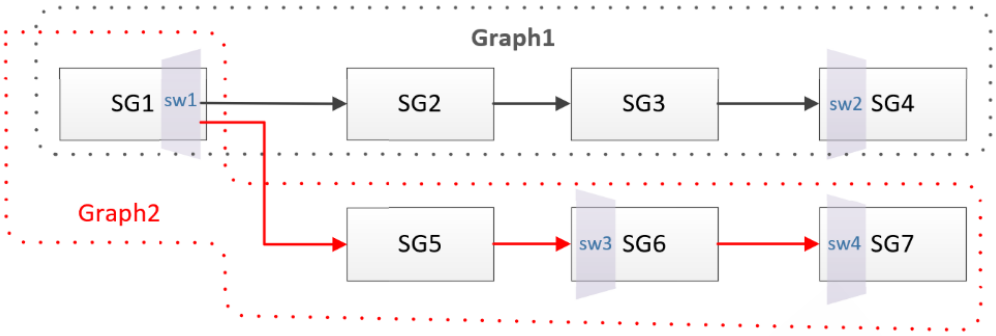
- A switch can be added into a subgraph (subgraph is parent of switch), or outside of any subgraph (independent switch). EC switch cannot be added in a subgraph.
- A switch can be connected from/to a module or another switch.

QACT uses switches for following purposes:

- Route connected subgraphs to create a graph automatically from end to end.
- Create GKV for a graph by combining all port KVs of switches in the graph.

In following example, two graphs (Graph1 and Graph2) can be created automatically based on connections between subgraphs and switches.

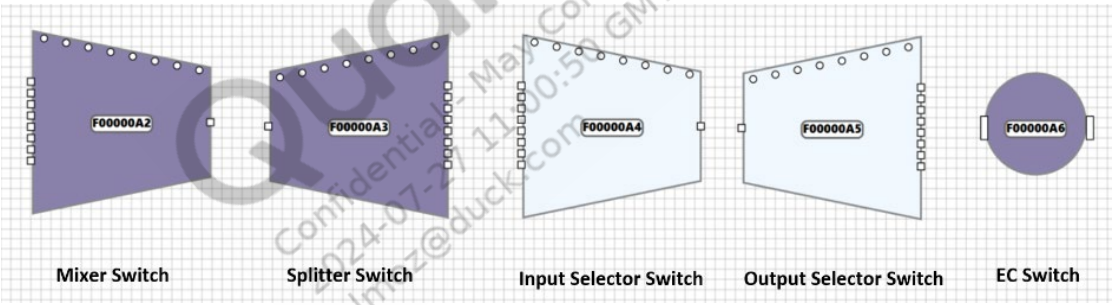
- Graph1: SG1 -> SG2 -> SG3 -> SG4. Its GKV is created by combining port KVs among sw1 and sw2 based on connections.
- Graph2: SG1 -> SG5 -> SG6 -> SG7. Its GKV is created by combining port KVs among sw1, sw3 and sw4 based on connections.



4.2.2.1 Switch types

There are five supported switches:

- Mixer switch – Concurrent N to 1 routing switch
- Splitter switch – Concurrent 1 to N routing switch
- Input selector switch – Non-concurrent N to 1 routing switch
- Output selector switch – Non-concurrent 1 to N routing switch
- EC switch – N to M routing switch. Only for EC graphs.



4.2.2.2 Switch port

A switch can have data ports (input and output) and control port (bidirectional). Each data port can only have one connection externally and internally except EC switch, in which one input/output port can be connected to multiple output/input ports.

For N to 1 Switch (Mixer and Input Selector), there are multiple input ports and one output port. The number of input ports are configurable in the switch's property view.

For 1 to N Switch (Mixer and Input Selector), there are one input port and multiple output ports. The number of output ports are configurable in the switch's property view.

Each switch port can have a Key Vector assigned in Key Configurator.

Control ports are used to connect control links between an internal module and a module outside of switch. Control ports are bidirectional .

The number of switch ports and control ports are configurable in Property view.

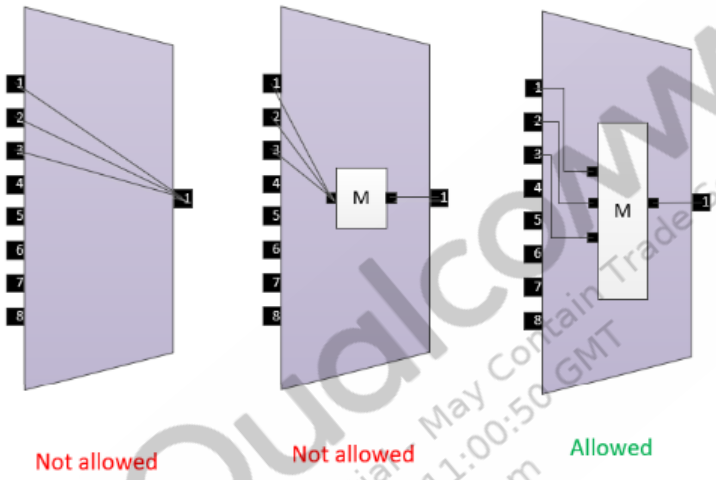
Same as module ports, switch input port IDs are even numbered, and output port IDs are odd numbered.

4.2.2.3 Inside switch

A switch can have internal modules added if the switch is inside a subgraph. For independent switches, no module can be added internally.

4.2.2.4 Concurrency

For concurrent switches (Mixer and Splitter), there are some limitations to avoid issues in run time. Inside a concurrent switch, there must be a module with concurrent capability, such as the Splitter module (in Splitter Switch) and Mixer module (in Mixer Switch). The following are examples for Mixer Switch concurrency.



For non-concurrent switches, there is no limitation.

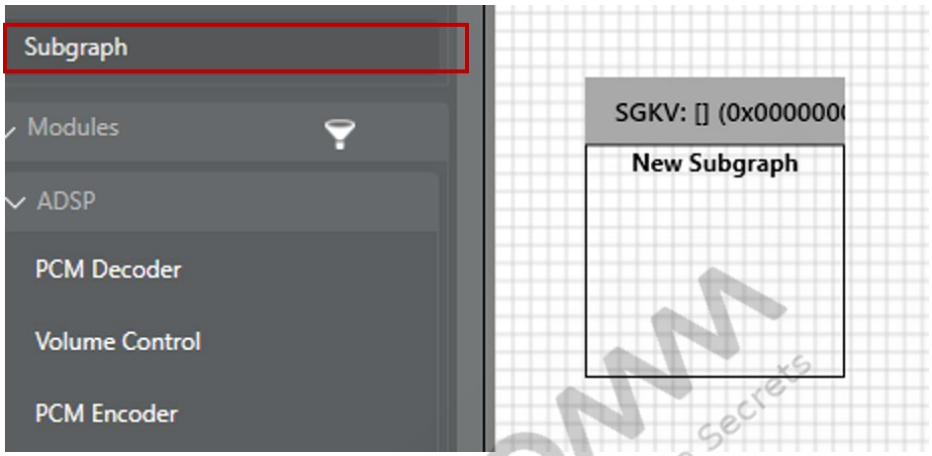
4.2.3 Graph modifications

To modify graph(s), click **Start Graph Modification** to start a modification session. Once this button is clicked, the user cannot select different use cases until graph modification is stopped.

4.2.3.1 Add subgraph

To add a subgraph:

1. Drag and drop Subgraph from the Module List View to Graph View.



2. QACT automatically assigns a subgraph ID and creates an empty subgraph.

4.2.3.2 Delete subgraph

To delete a subgraph:

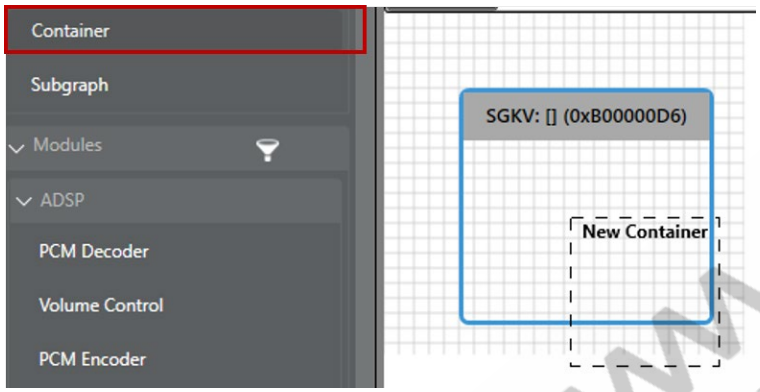
1. Select the subgraph that needs to be deleted.
2. Press the **Delete** key.

All the containers, modules and connections associated with the subgraph are deleted. Additionally, the GKV(s) that contain this subgraph are modified by removing the SGKV.

4.2.3.3 Add container

To add a container:

1. Drag and drop Container from the Module List View to the intended subgraph.



2. QACT automatically assigns a container ID and creates an empty container.

In the case that the new Container is dropped onto an empty Graph View, QACT automatically creates an empty subgraph and adds the container inside it.

4.2.3.4 Delete container

To delete a container:

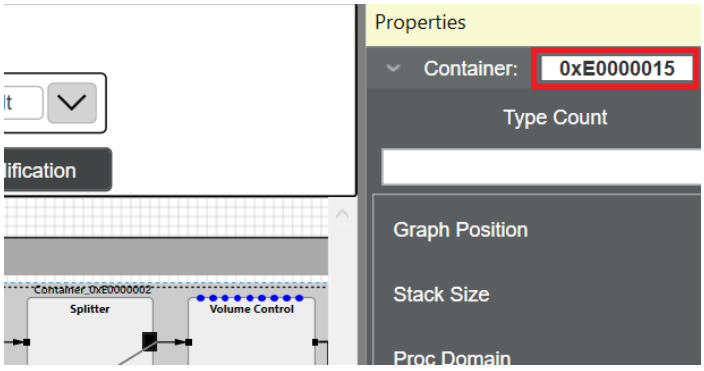
1. Select the container that needs to be deleted.
2. Press the **Delete** key.

All the modules and connections present in the container are also deleted in the process.

4.2.3.5 Reuse a container across subgraphs

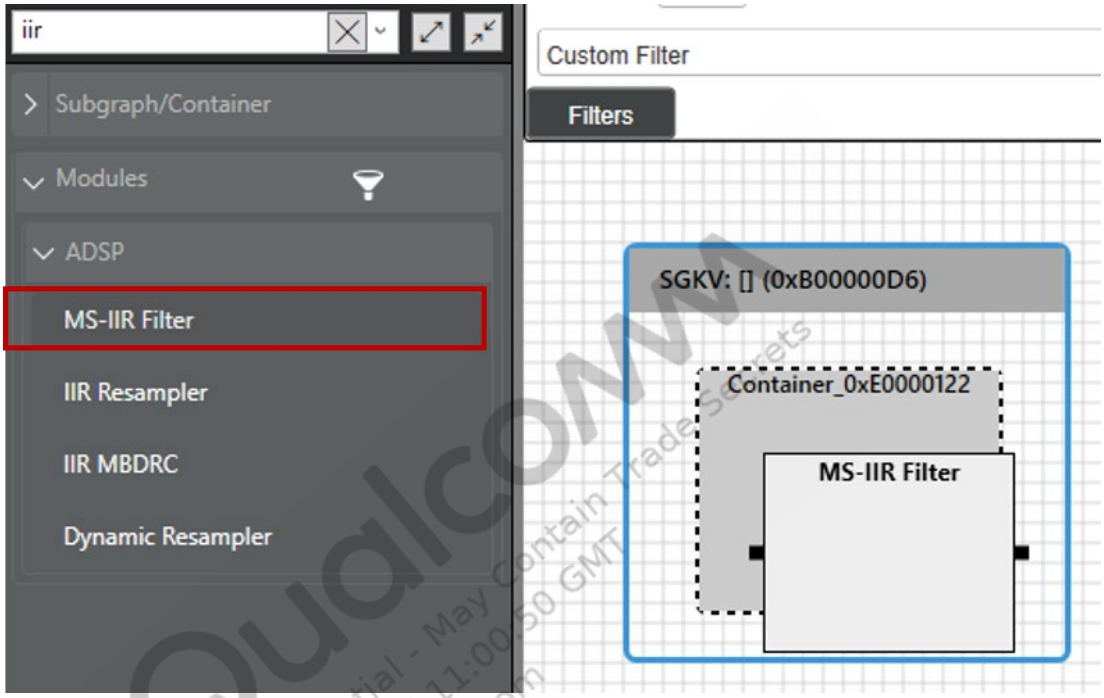
To reuse a container across subgraphs:

1. Click on **Start Graph Modification** button.
2. Select the container that needs to be reused.
3. In properties view, change the container ID accordingly.



4.2.3.6 Add module

1. Select a module to be added into a subgraph/container. Users can search for the module with its name to filter the list as well.
2. Drag and drop the module from Module List View to the intended container.



- In the case that the module is dropped onto an empty Graph View, QACT automatically creates an empty subgraph and container and places the module inside it.
- In the case that the module is dropped onto a subgraph, QACT automatically creates an empty container and places the module inside it.
- Default data is created for the modules as well when the module is dropped.

NOTE: For a module to be added into a container, the container type and proc domain should support the type of module being added. Otherwise, QACT shows an error that the module cannot be added.

4.2.3.7 Delete module

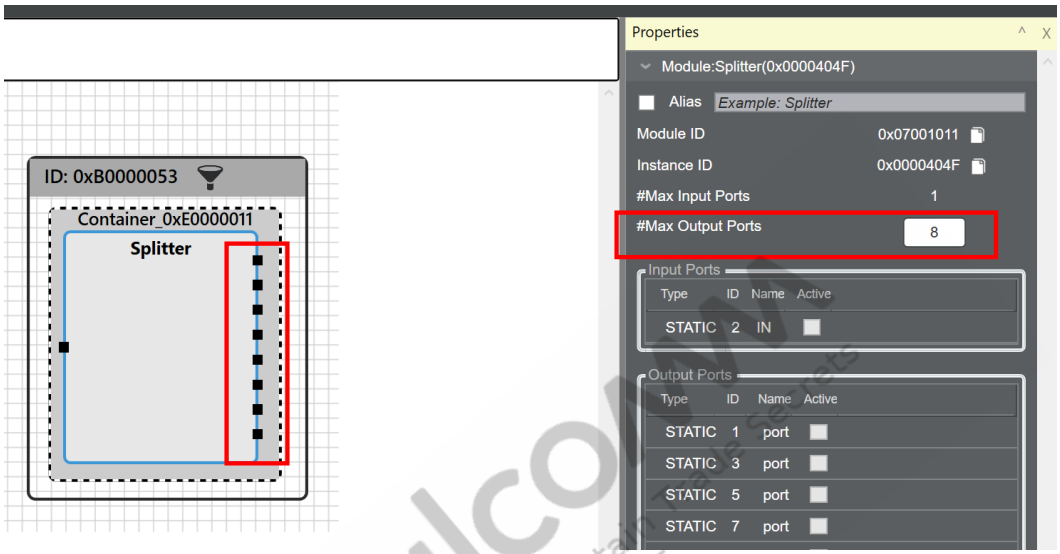
To delete a module:

1. Select the module that needs to be deleted.
2. Press the **Delete** key.

All the connections to/from module are also deleted in the process.

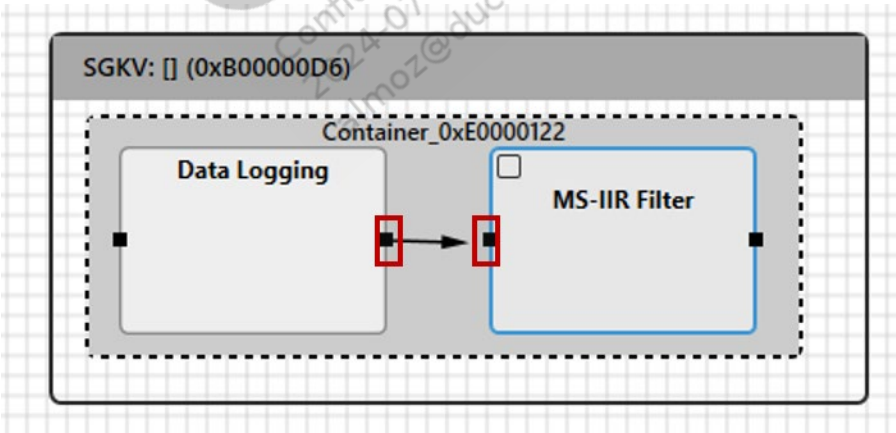
4.2.3.8 Configure module's dynamic ports

Some modules have dynamic input/output ports. The number of dynamic ports for such a module can be adjusted in module's property view. For modules without dynamic ports, the Max Input/Output Ports field in the Properties window is not editable.



4.2.3.9 Add data link

To add a data link, click the output port of the source module and drag onto the input port of the destination module.



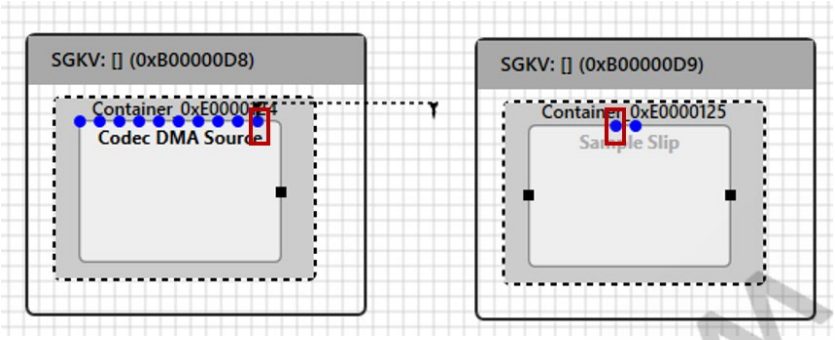
4.2.3.10 Remove data link

To remove a data link:

1. Select the data link to be deleted.
2. Press the **Delete** key.

4.2.3.11 Add control link

To add a control link, click the intent of the source module and drag onto the intent of the destination module.



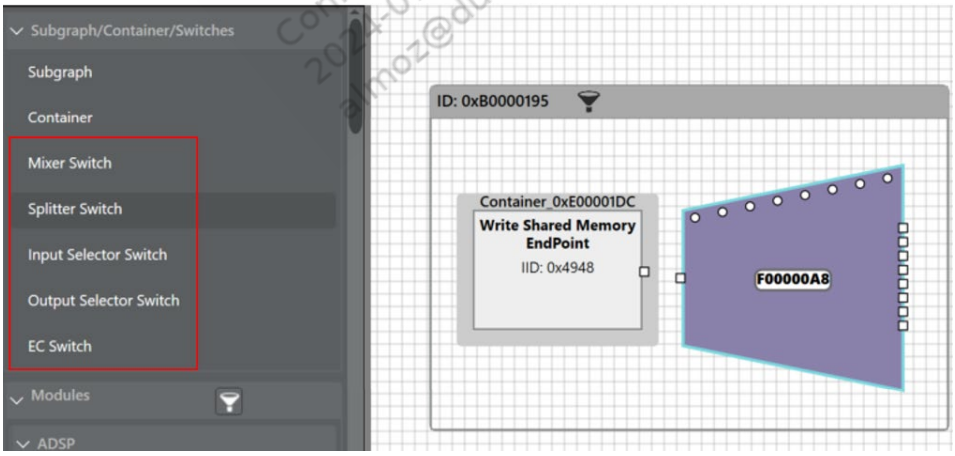
4.2.3.12 Remove control link

1. Select the control link to be deleted.
2. Press the **Delete** key.

4.2.3.13 Add switch

To add a switch:

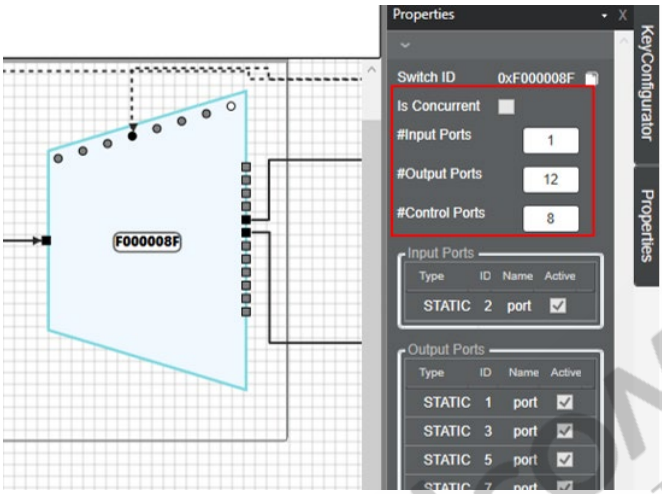
1. Select a switch from the switch list.
2. Drag and drop it into a subgraph or a blank area of graph view.



4.2.3.14 Configure switch

4.2.3.14.1 Switch properties

Click a switch to view and configure its properties.

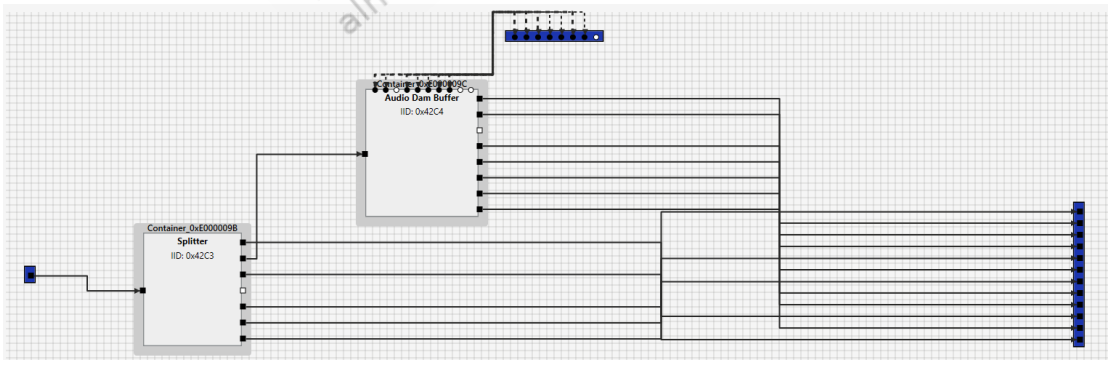


The Input Ports, Output Ports, and Control Ports fields are used to set the number of ports of a switch.

The Is Concurrent checkbox indicates whether a switch supports concurrency or not. It can be updated. However, while changing a non-concurrent switch to concurrent, it maybe not allowed after checking concurrency limitation introduced in Section 4.2.2.4.

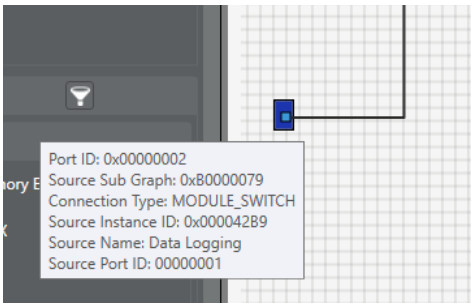
4.2.3.14.2 Switch Configurator

Double click a switch to view and configure its internal modules, connections, and control links.



Inside a switch, module and container properties can be configured in Property View as well. Modules and containers in a switch are also inside the switch's parent subgraph in ACDB.

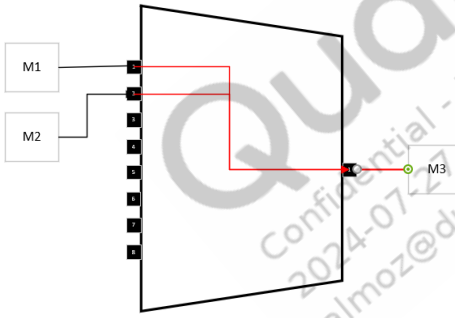
Hover the cursor over a port to see its information as shown in the following diagram. For an input port, if it is connected, the connection's source information will be shown. For an output port, if it is connected, the connection's destination information will be shown.



4.2.3.15 Switch Connection

While updating an input or output connection of a switch, QACT can do some automatic work and safe check.

For a switch without an internal module, adding an external input/output connection to a switch port will automatically add internal connection(s) for the same port if the switch has output/input connection already. In following example, if M1 and M2 already connect to a switch's input ports, while adding an output connection to M3, QACT will automatically add two internal connections from the two input ports to the output port. For a switch with internal module(s), users must manually connect switch ports with the module(s) firstly, then add external connection(s) as needed.



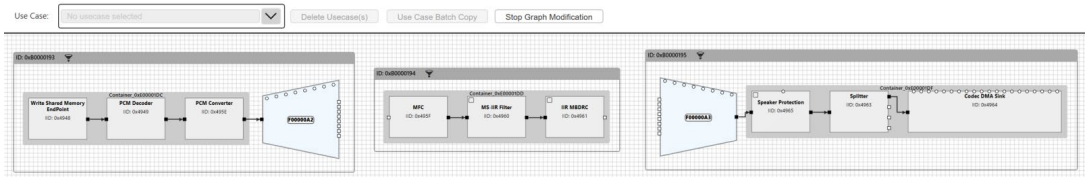
4.2.4 Create a new graph

4.2.4.1 Create a new graph by adding subgraphs from scratch

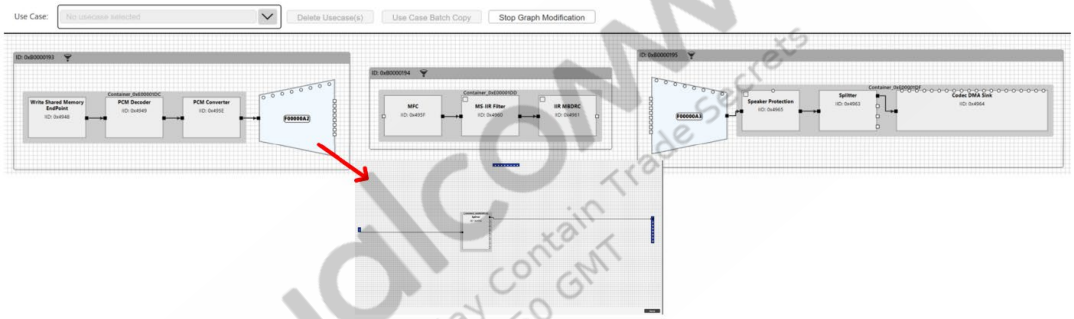
To add subgraphs from scratch:

1. Click **Start Graph Modification**.
2. Drag and drop modules, containers, and subgraphs into Graph Designer and connect modules inside subgraphs. Do not add connections between subgraphs.

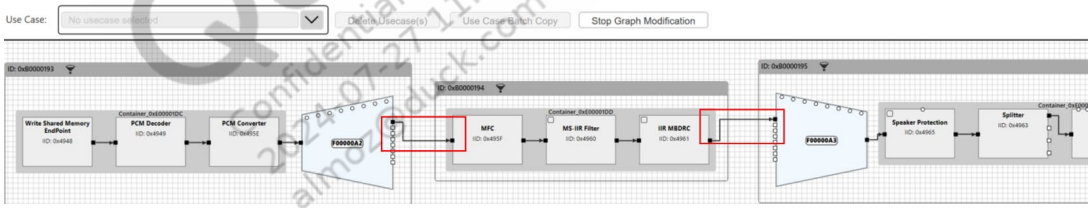
3. Add switch(es) into Graph Designer.
   - A switch can be added inside a subgraph or outside of a subgraph.
   - A switch must be added at the end of a subgraph or the beginning of a subgraph (not the middle of subgraph).



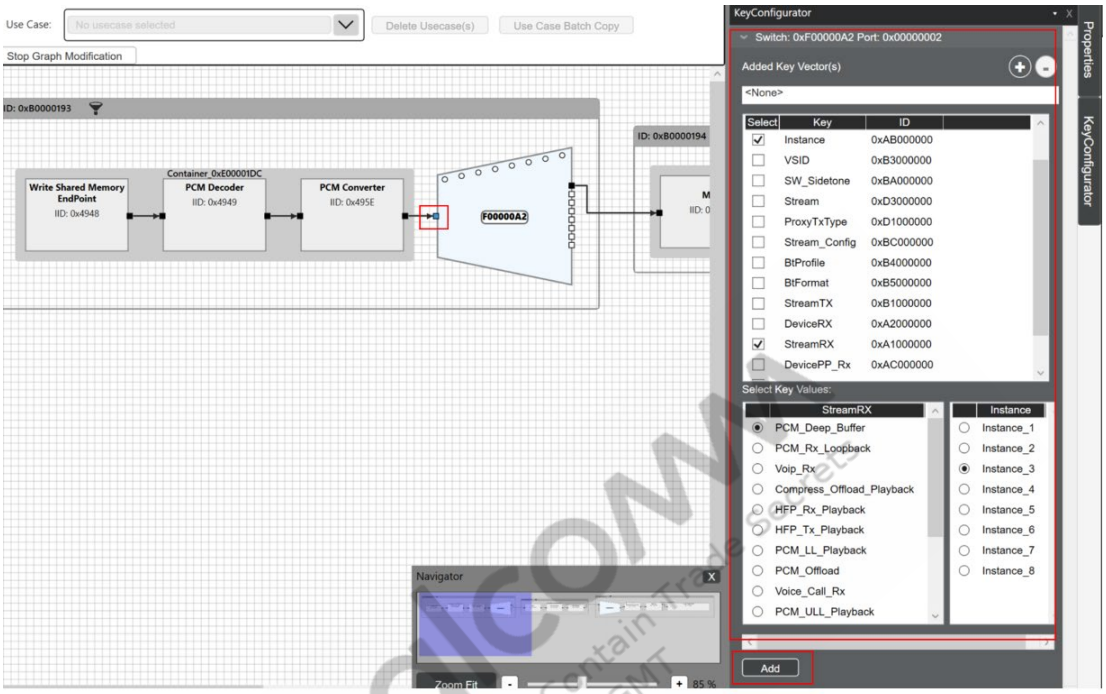
4. For each switch, if an internal module is needed, double click it to add the module in the switch, connect it with the switch input port and output port, and click **Done**. If needed, a switch internal module also can have control link connected internally.



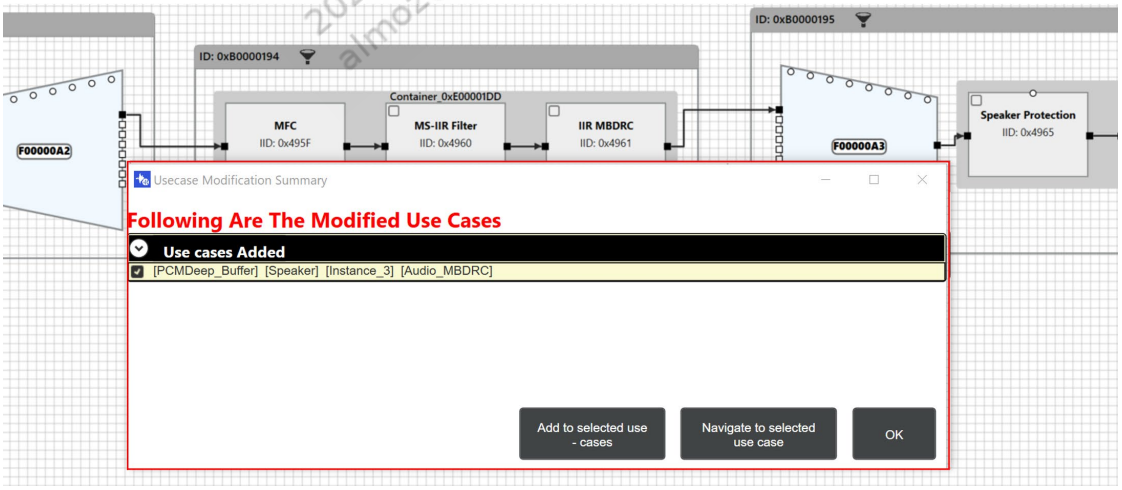
5. Connect switches between subgraphs.



6. For each connected switch port on this graph, click it to assign a port Key Vector in Key Configurator on the right panel.



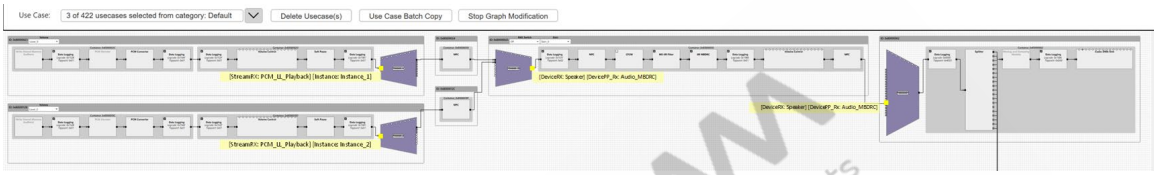
7. Click **Stop Graph Modification** to finish adding a new graph. A summary dialog is shown to indicate a new GKV is added. To view the new added graph, select it and click **Add to selected use cases** to show the graph with other selected graphs in graph view. Or, click **Navigate to selected use case** to only show the newly added graph without any other graphs. Click **Ok** to close it. The new added GKV will be shown in the Use Case drop-down box.



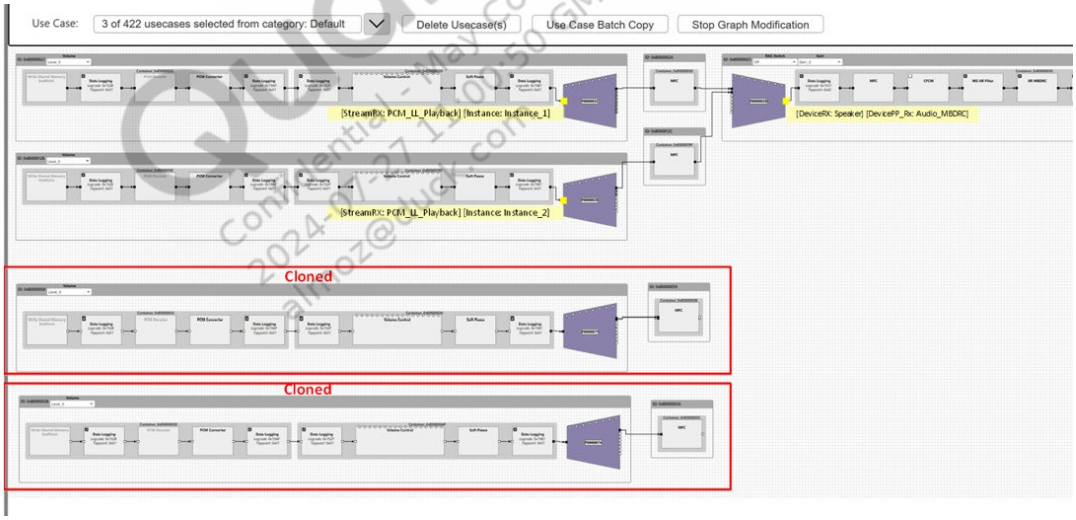
4.2.4.2 Create new graph(s) based on existing graphs

Since QACT v8.1 can automatically create graphs for connected subgraphs, connecting new added subgraph(s) to other existing subgraph(s) can have new graph(s) created automatically. For example, for a given Audio Playback use case, there are a Stream subgraph with instance1 and Speaker subgraph. If a new Stream subgraph with a new instance is added and connected to the same Speaker subgraph, a new Audio Playback with a new Stream instance will be created.

1. In the Use Case drop-down box, select existing GKV(s) (for example, Audio Playback GKVs).



2. Click **Start Graph Modification**. Create new subgraphs by either dragging modules or cloning existing subgraphs.
   a. To clone a subgraph, right click it and select **Clone**.
   b. Newly cloned subgraphs will have everything cloned from original subgraph, including switch and switch port Key Vectors.



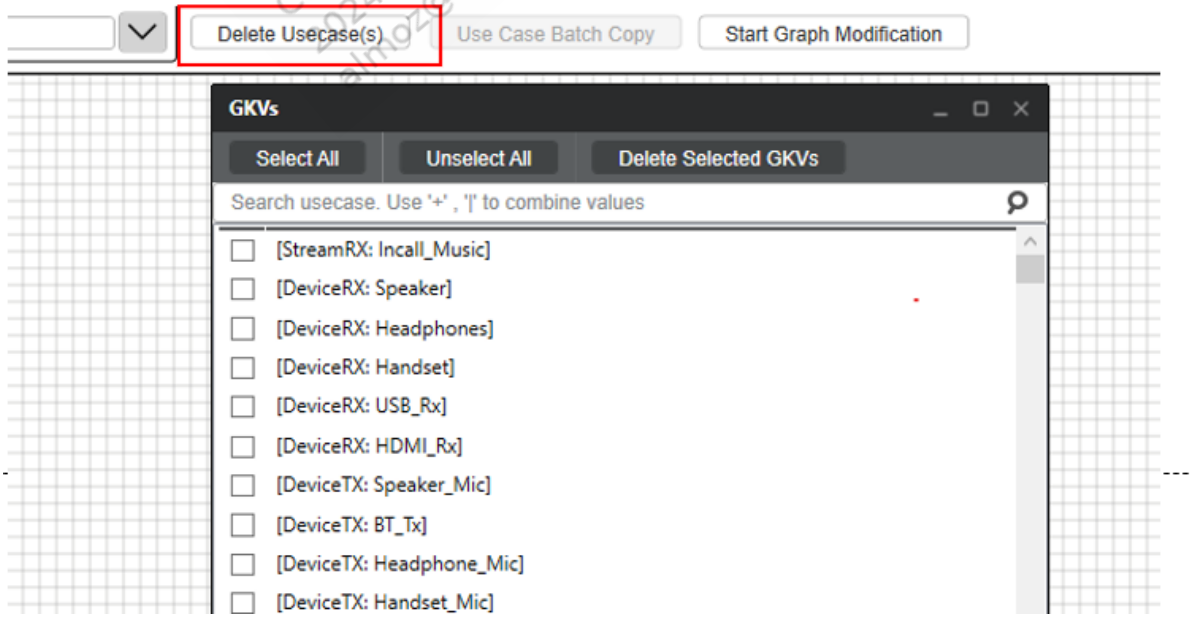
3. Connect new subgraphs to switch ports in existing subgraph.

4. For newly connected switch ports, assign a port Key Vector properly. For example, new switch ports should have new StreamRX and Instance Key Vector added ([StreamRX: PCM_LL_Playback][Instance: Instance_3] and [StreamRX: PCM_LL_Playback][Instance: Instance_4]).

5. Click **Stop Graph Modification**. A summary dialog will show two new GKVs added.
6. Close the summary dialog. The newly added GKVs will be added in the Use Case drop-down box.

4.2.5 Delete a graph

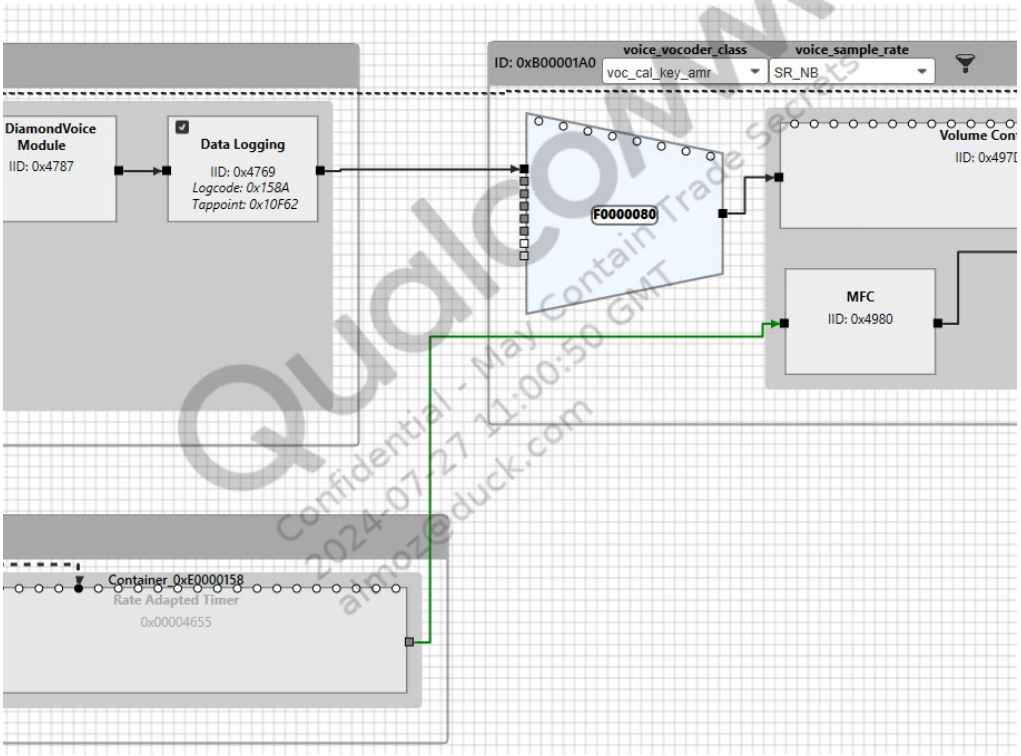
To delete a graph (use case), click **Delete Usecase(s)**. A dialog will be launched. GKV(s) can be selected and deleted.



NOTE: After a graph is deleted, its subgraphs will not be deleted automatically. If a subgraph is being used by other graph(s), do nothing for the subgraph. If a subgraph is not being used in any graph, move it to the zero GKV.

4.2.6 Dangling links

A dangling link is added between two subgraphs which belong to two different graphs. A dangling link can be a data connection, or control link. To add a dangling link, right click a module's data port or control port, select **Add Dangling Connection**. Next, select another module port from different graph and select **Complete Dangling Connection**. A dangling link will be added. To differentiate a dangling link from other connections, it will be shown as green colored. A switch with internal module(s) also can have dangling link added.



4.3 Switch and routing policies

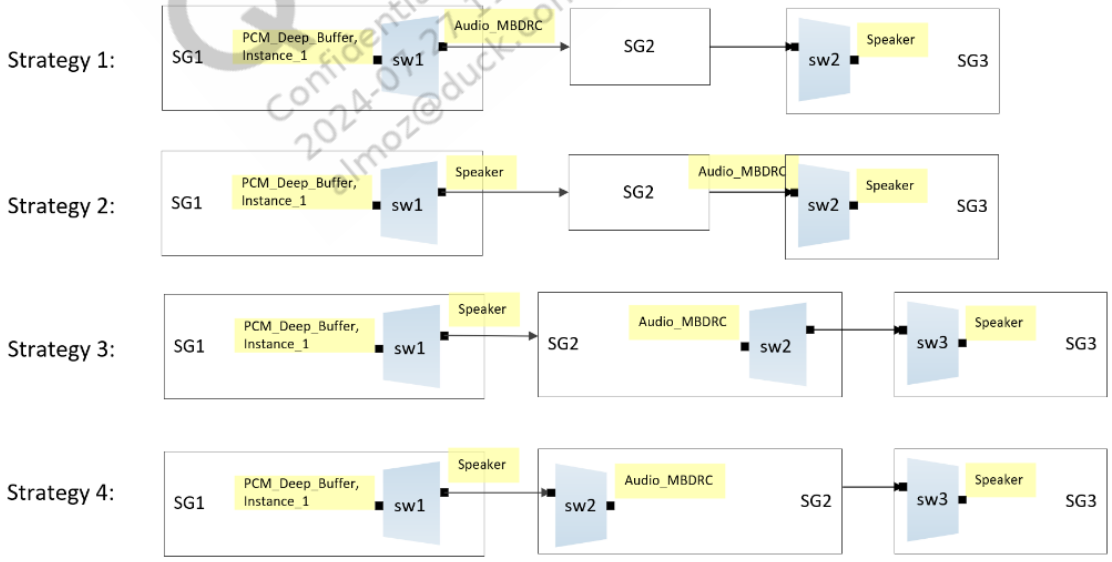
While stopping graph modification, QACT uses added/deleted/modified switches and subgraph connections to update use cases automatically based on some routing policies.

4.3.1 Policies to add switches in graph

While designing use cases, switches are typically added at the start, end, or both positions of a subgraph, so that QACT can use them to route to create use cases.

- A 1 to N Switch (Splitter and Output Selector) only can be added at the end of a subgraph. This means its input port only can be connected from a module in the same subgraph. Its output port cannot have a connection to a module in the same subgraph – It must connect to a module in another subgraph. Otherwise, QACT will show an error.
- A N to 1 Switch (Mixer and Input Selector) only can be added at the beginning of a subgraph. This means its input port cannot have a connection from a module in the same subgraph – It must be from a module in another subgraph. Its output port only can be connected to a module in same subgraph. Otherwise, QACT will show an error.

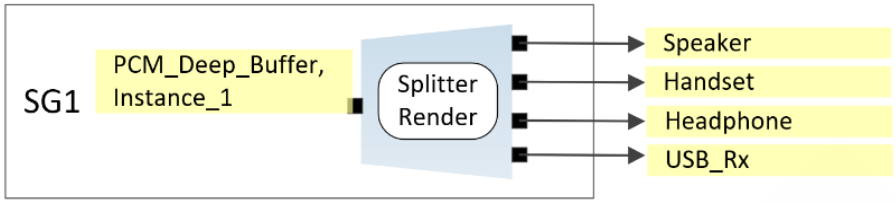
For one use case, there can be multiple ways to add switches and assign switch port KVs to get the same GKV and graph data. For example, in the following use case, all strategies will create exactly the same GKV (*PCM_Deep_Buffer + Instance_1 + Audio_MBDRC + Speaker*) and graph data, although their switch positions and KV assignments are different. All of these strategies will work end-to-end. To decide which one to be used, the user must consider how a subgraph will be shared with other graphs. In the following graphs, SG2 is DevicePP_RX. Considering it can be connected from different StreamRX subgraphs from the upstream side, strategy 4 is ideal. Especially, in SG2, the first module is typically Sync Module, which should be added in sw2 as well.



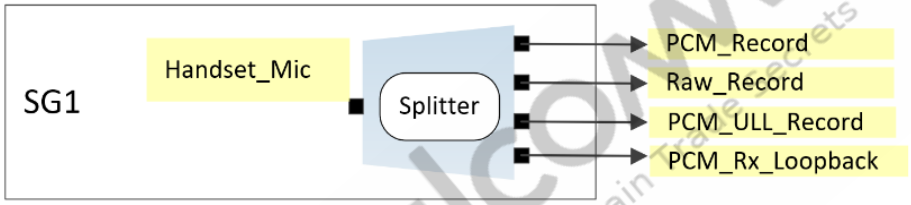
4.3.1.1 Switch in a source end point subgraph

Typically, a source end point subgraph can be connected to multiple downstream subgraphs. A 1 to N type switch (Splitter or Output Selector) should be added at the end of the subgraph so that all graphs connected from this subgraph will be created by routing this switch's input port and output port KVs. The following are examples for this scenario.

For Audio Playback use cases, in the StreamRX subgraph, a Splitter or Output Selector switch can be added at the end of the subgraph. A Splitter Render module can be added inside the switch to route the stream instance to all its downstream subgraphs (StreamPP, DeviceRxPP, DeviceRx etc.).



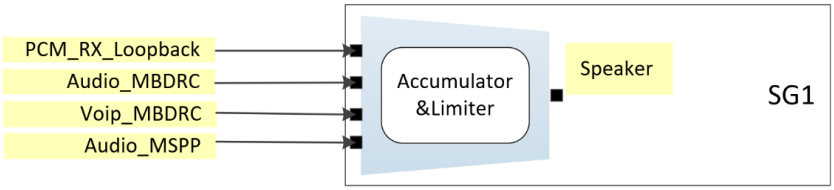
For Audio Recording use cases, in the DeviceTX subgraph, a Splitter or Output Selector switch can be added at the end of the subgraph. A Splitter module can be added inside the switch to route device Tx to all its downstream subgraphs (DeviceTxPP, StreamTx etc.).



4.3.1.2 Switch in a sink end point subgraph

A sink end point subgraph can be connected from multiple upstream subgraphs. A N to 1 type switch (Mixer or Input Selector) should be added at the beginning of the subgraph so that all graphs connected to this subgraph will be created by routing this switch's input port and output port KVs. The following are examples for this scenario.

For Audio Playback use cases, in the DeviceRX subgraph, a Mixer or Input Selector switch can be added at the beginning of the subgraph. An Accumulator&Limiter module can be added inside the switch to route all its upstream subgraphs (StreamPP, DeviceRxPP, StreamRx etc.).



4.3.1.3 Switch in a middle subgraph

If a subgraph has both upstream and downstream subgraphs connected in a use case, there are multiple options to add switch(es) in it. Such a subgraph may have a switch in beginning, or end, or both, or none. It also depends on how its upstream and downstream subgraphs have switches added.

Examples are shown in the diagram in Section 4.3.1, in which SG2 is a middle subgraph.

4.3.2 Independent switch

An independent switch means a switch does not belong to any subgraph. It cannot have any internal module added.

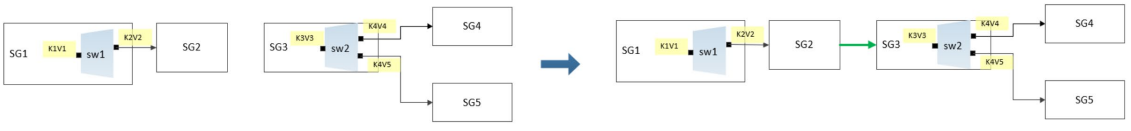
An independent switch can be helpful if the system designer prefers graphs with multiple levels routing visually (although such graphs can be created by using different strategies to assign switch port KVs in most cases). For example, the following two designs will create exactly the same four use cases.

4.3.3 Graphs are added/deleted at stop graph modification

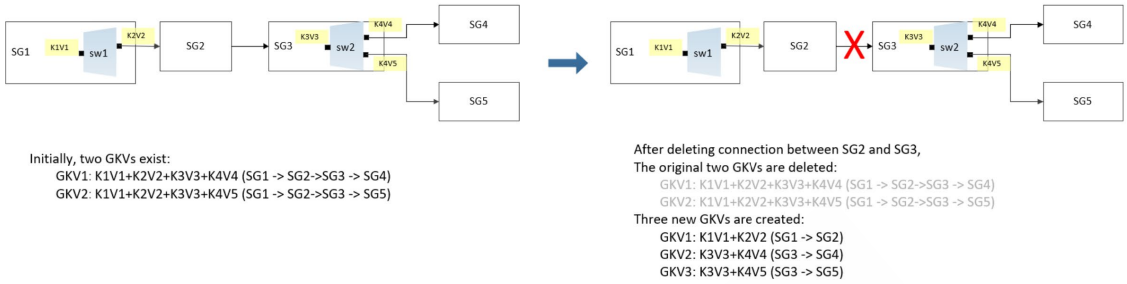
After **Stop Graph Modification** is clicked, QACT will automatically update graphs based on modified routings (connections) and switch port KVs. Some graphs may be added and some graphs can be deleted. This procedure is handled based on following policies:

- Only linear connected/routed subgraphs will be considered as one graph.
- A routed graph starts from a source end point subgraph and ends with a sink end point subgraph. Here, source EP and sink EP are not subgraph properties, but subgraph positions.
- A linear routed graph's GKV is automatically created by combining KVs of all switch ports on that route.
- If a graph has one connection deleted, it can be broken into two graphs. One graph contains all connected subgraphs on the upstream side of the deleted connection. The other one contains all connected subgraphs on the downstream side of the deleted connection. However, if the upstream adjacent subgraph of deleted connection has other downstream connections, there will be no new graph created for the upstream side. If a downstream adjacent subgraph of a deleted connection has other upstream connections, there will be no new graph created for the downstream side.
  - The same logic applies for deleting a subgraph scenario as well.

The following diagram illustrates the concept that adding a new connection between two subgraphs can create new graphs.



The following diagram illustrates the concept that deleting a connection between two subgraphs may delete graphs and add new graphs.



4.3.4 Graph validation

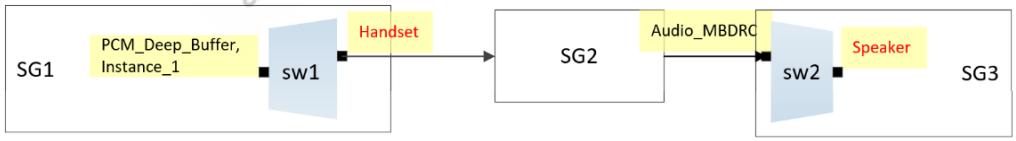
Once **Stop Graph Modification** is clicked, QACT automatically updates graphs based on routing policies. It then automatically validates graph data and switch data. If any error or warning is found, it is shown in validation manager view (refer to Section 4.12). The system designer should fix errors and warnings before doing further updates to avoid corrupted data.

4.3.4.1 Graph data validation

For graph data, following errors and warnings can be validated and reported.

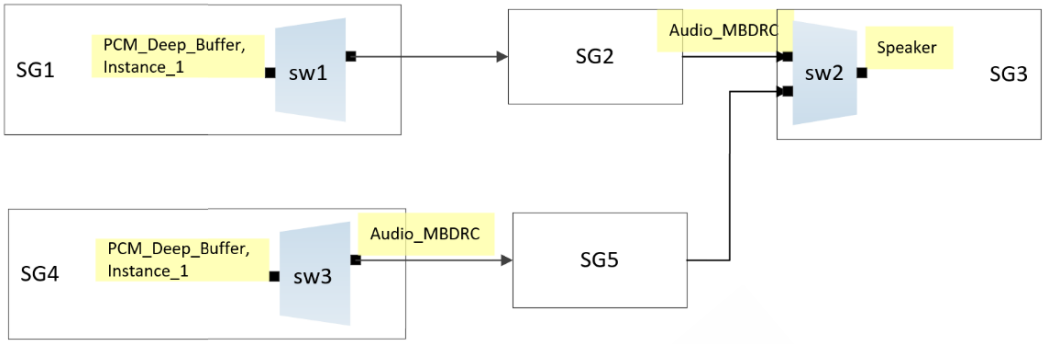
- E130: Invalid GKV. A graph has conflict KVs. Its subgraphs are assigned with zero GKV.
  - This error can be caused by a routed graph's switches have conflict kvs. User should update switch port KV(s) to solve this issue.

  Example: in following routed graph, sw1's output port KV and sw2's output port KV are conflict. One is "Handset". The other one is "Speaker".



- E131: Duplicate GKV. Some graphs have same GKV.
  - While routing graphs to create GKVs, different graph have same GKV by checking their switch port kvs. In this case, all these graphs are assigned to one GKV.
  - If some graphs have conflict GKV detected (E130), these graphs are assigned to zero GKV. This error will also be reported since multiple graphs have same GKV (zero).

  Example: in following diagram, there are two routed graphs are detected (SG1->SG2->SG3 and SG4->SG5->SG3). By routing their switch port KVs, both graphs have same GKV: PCM_Deep_Buffer + Instance_1 + Audio_MBDRC + Speaker.



- W101: warning that no endpoint is found in this graph. Either source or sink endpoint subgraph is missing in this graph.
  - An endpoint subgraph means it starts from a module without input port (source endpoint) or ends with a module without output port (sink endpoint).
  - This is just a warning to remind user to make sure such a graph is valid or not. It can be ignored if it is valid.
- W102: a graph's GKV is zero GKV. This is a warning. Typically, zero GKV is considered as a invalid graph. It cannot be used on device. User should either fix it or delete it.
  - Sometimes zero GKV can be created when a graph has invalid/conflict GKV detected (E130).
- W103: Warning that a manually created graph contains more than one subgraphs but some subgraph pair is missing. To fix it, the user should delete this GKV, select all subgraphs and connections between them, and then manually create the GKV again (refer to Section 4.4).

4.3.4.2 Switch data validation

For Switch data, following errors and warnings can be validated and reported.

- E132: an input or output port has no internal connection but it has external connection.
  - This error has to be fixed. Otherwise, routed graph will not have correct connections. For example, a switch is connected from an upstream module1 to its input port. And connects to a downstream module2 from its output port. For the routed graph, there should be a connection between module1 and module2. If there is no internal connection between the input port and output port, module1 and module2 are not connected in the graph.
- E133: Switch's input port and output port have conflict port kvs.
  - While routing switches and subgraphs to create GKV(s), switch input and output port kvs are used. If a switch input and output port kvs are conflict, the routed graph cannot have correct GKV created. For example, if a Mixer Switch's input port 1 has KV: DeviceRX: Speaker, and its output port KV is DeviceRX: Handset, there is no way for QACT to know which DeviceRX value should be used. User has to correct port kvs.
- E134: a Switch has invalid internal connection for its concurrency.

- This error is typically for a concurrent switch since a non-concurrent switch has no limitation for internal connections (refer to Concurrency).

- E135: a Switch has internal or external control link missing, or intents do not match with connected module control port.
  - If a switch's control port has an internal connection to its internal module, but no external connection, this error is reported.
  - If a switch's control port has an external connection to other module, but has no internal connection, this error is reported.
  - If a switch's control port has a connection, but its intent(s) is different than the connected module's control port intent(s), this error is reported.
- E146: an independent switch is not used in any graph.
  - Such a switch is useless and redundant. User can click AutoFix to delete it or connect it to some subgraph.
- E147: a switch has no input/output external connection.
  - A switch should at least have one input connection and output connection. Otherwise, it is not used in any routing path.
- E148: a switch's input or output port has more than one external connections.
  - A switch port only can have one external connection.
- E149: a switch has internal module(s), but it has internal connection directly from input port to output port.
  - If a switch has internal module(s), means all its internal connections should use the module(s). It is not allowed to have an internal connection from an input port to an output port directly.

4.3.5 EC switch

EC switch is a special switch that should only be used in EC use cases only. As long as a graph contains an EC switch, that graph is considered as an EC use case. Typically, an EC use cause starts from a RX device subgraph, and connects to one TX subgraph through an EC switch.

EC switch has following special policies than other switches:

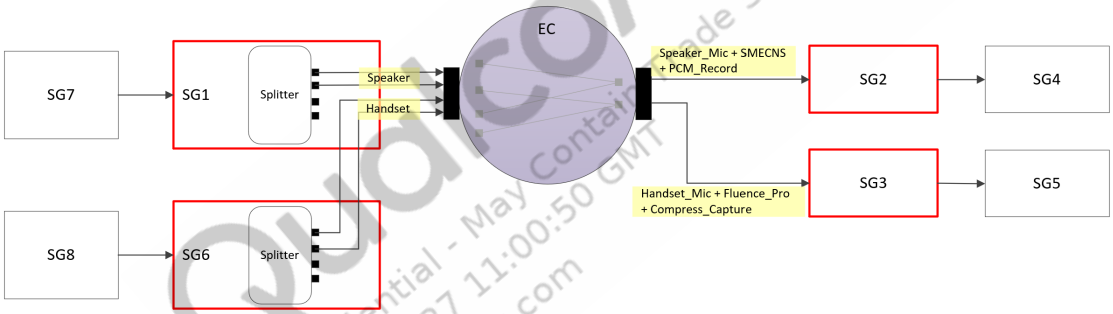
- EC switch supports multiple input ports and multiple output ports. Internally, one input port can connect to multiple output port. One output port can connect from multiple input ports.
- EC switch only can be used as an independent switch outside of subgraphs. It cannot have any internal module.
- EC switch port KVs can be added on input connection and output connection instead of ports directly.
- When Stop Graph Modification is clicked, QACT will automatically route the first upstream adjacent subgraph of EC switch to its downstream subgraph to create one EC use case. This means the automatically created EC graph will only contain two subgraphs through EC switch.

- An EC use case's GKV is created by combining Ethe C switch's input KV and output KV. All other switches in the graph will not be considered for EC GKV.

The following are examples of EC use cases created by routing EC switch. Four EC use cases will be automatically created:

- Use case 1: Speaker+Handset_Mic+FluencePro+PCM_Record contains SG1->SG2
- Use case 2: Speaker+Handset_Mic+FluencePro+Compress_Capture contains SG1->SG3
- Use case 3: Handset+Handset_Mic+FluencePro+PCM_Record contains SG6->SG2
- Use case 4: Handset+Handset_Mic+FluencePro+Compress_Capture contains SG6->SG3

Although SG7 and SG8 are connected to SG1 and SG6, they will not be included in any EC use case. Although SG4 and SG5 are connected to SG2 and SG3, they will not be included in any EC use case.

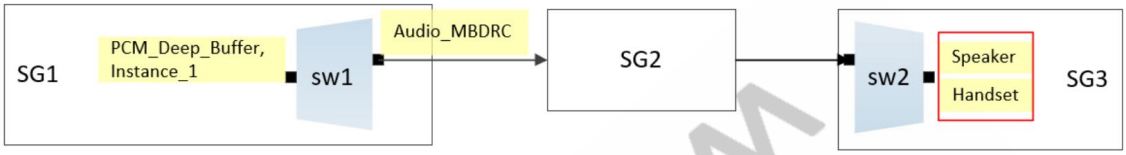


4.3.6 Multiple KVs for one switch port

One switch port can have more than one KVs added if needed. When a port has multiple KVs, all graphs routing through this port will have multiple GKVs created.

In the following example, SW3's output port has two KVs: Speaker and Handset. Then two GKVs with the same subgraphs will be created:

- GKV1: PCM_Deep_Buffer+Instance_1+Audio_MBDRC+Speaker
- GKV2: PCM_Deep_Buffer+Instance_1+Audio_MBDRC+Handset

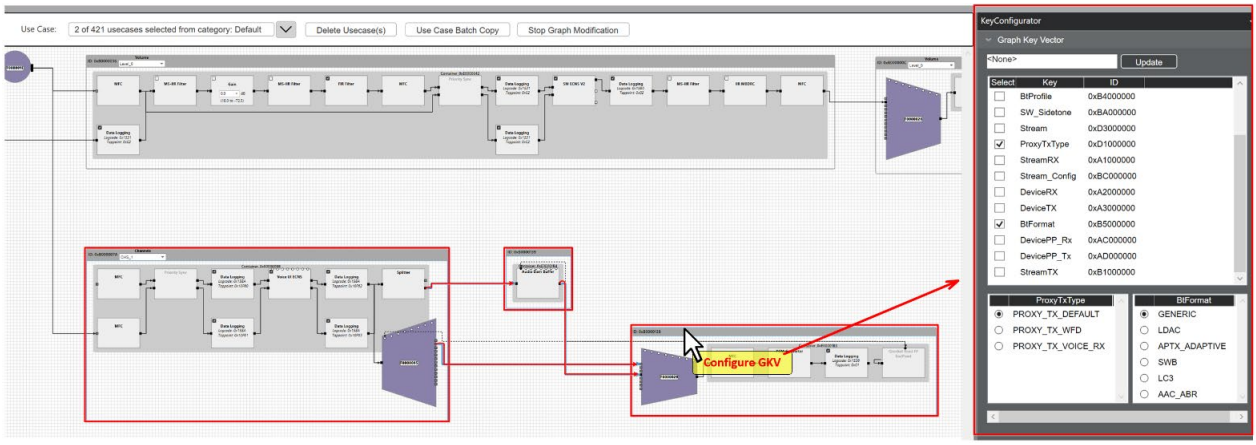


4.4 Manually generate a graph

QACT can automatically create graphs by routing connected subgraphs. However, in some scenarios, the system designer may want to design graphs by manually selecting some subgraphs. There are two typical scenarios:

- Graph only containing one single subgraph – If the system designer needs to design a graph that only contains one subgraph, manually generating the graph is the only way since a single subgraph is either not connected with any other subgraph (QACT cannot route it), or it will be always included in a graph with other connected subgraphs.
- Graph containing connected subgraphs which are partial of a graph – For example, a graph containing sg1->sg2->sg3->sg4 is created. If the system designer needs to create a graph only with sg2->sg3, the only way is to manually select the two subgraphs and connections between them to create a graph.

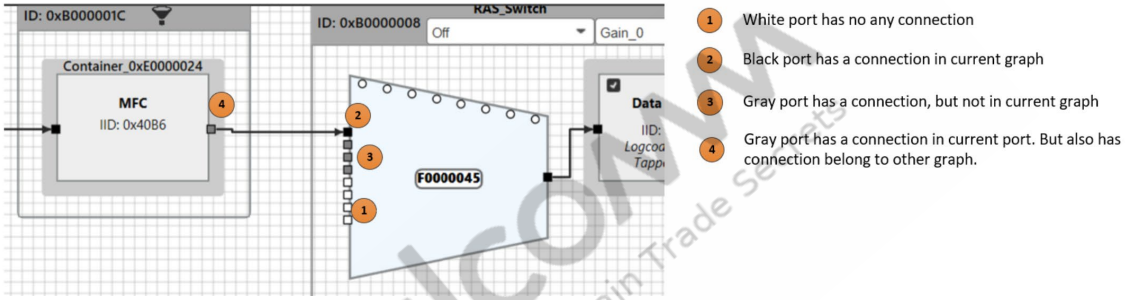
To create a manual graph, just multi-select the subgraph(s) and connections between them, then right click and select **Configure GKV** to add the GKV for the manual graph.



4.5 Port coloring

For selected graph(s), some switches and modules may have some connections to other graphs. Port coloring is a feature to indicate if a switch or module port has a connection or not. There are three colors for a port:

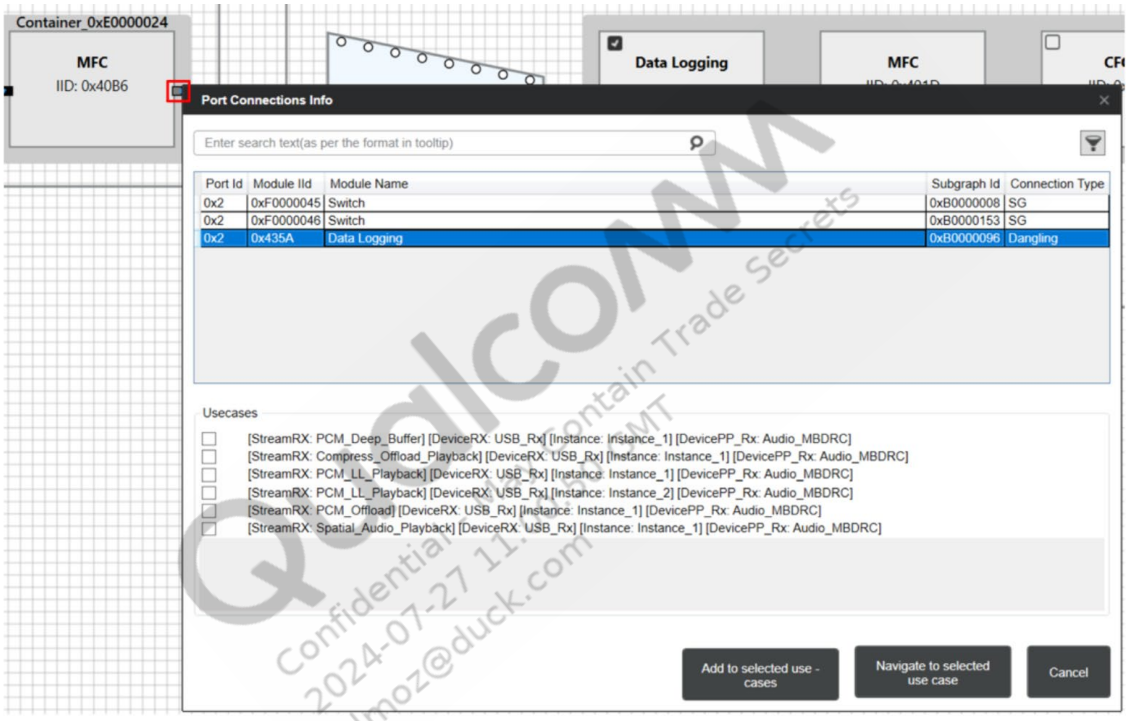
- White – The port has no connections.
- Black – The port has connection(s), all connections are part of the selected graph(s), and already shown in GraphDesigner view.
- Gray – The port has connections, but the connection is not part of any currently selected graphs.



For a gray port, the system designer can right click it and click **Show all Connections** to launch the Port Connection Info dialog. In the dialog, the other end information of connection to/from this gray port will be shown.

The following example shows that MFC's output port is gray, and right clicked to show all its connections. In the dialog, the upper section shows all connections from the selected MFC. The other end of connection can be a module or switch from another subgraph. The last column, "Connection Type", indicates the connection is SG (connection between subgraphs) or Dangling (dangling link).

Select one connection and all use cases/GKVs containing the connection will be shown in bottom section of the dialog. Click **Add to selected use cases** to add graph(s) for the selected GKV(s) to Graph Designer view. Click **Navigate to selected use case** to only show graph(s) for the selected GKV(s) to GraphDesigner view.

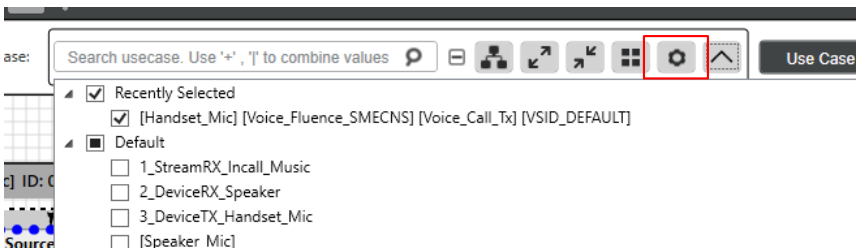


4.6 Use case categorization/alias

Use cases can be grouped as use case categories (for example, Fluence, Speaker, Playback, etc.) for ease of navigation. Use cases can also be assigned use case aliases.

To access Usecase Categorization View, do either of the following:

- Click **Edit -> Options**, or
- Click the settings button in the use case dropdown box



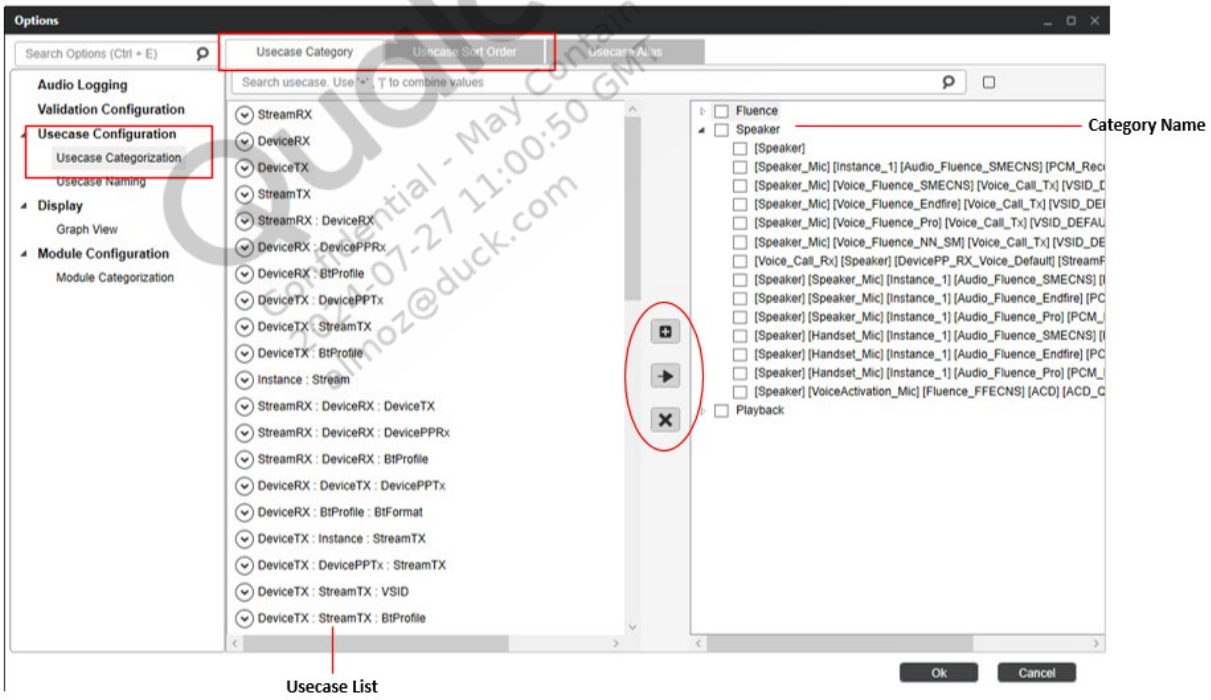
Then select **Usecase Configuration -> Usecase Categorization** view in the **Options** window.

4.6.1.1 Use case category

The Usecase Category tab displays a list of all the use cases, grouped by keys on the left side and a list of Usecase Categories on the right side.

The user can create custom use case categories and group use cases under these categories. A use case may be grouped under more than one category.

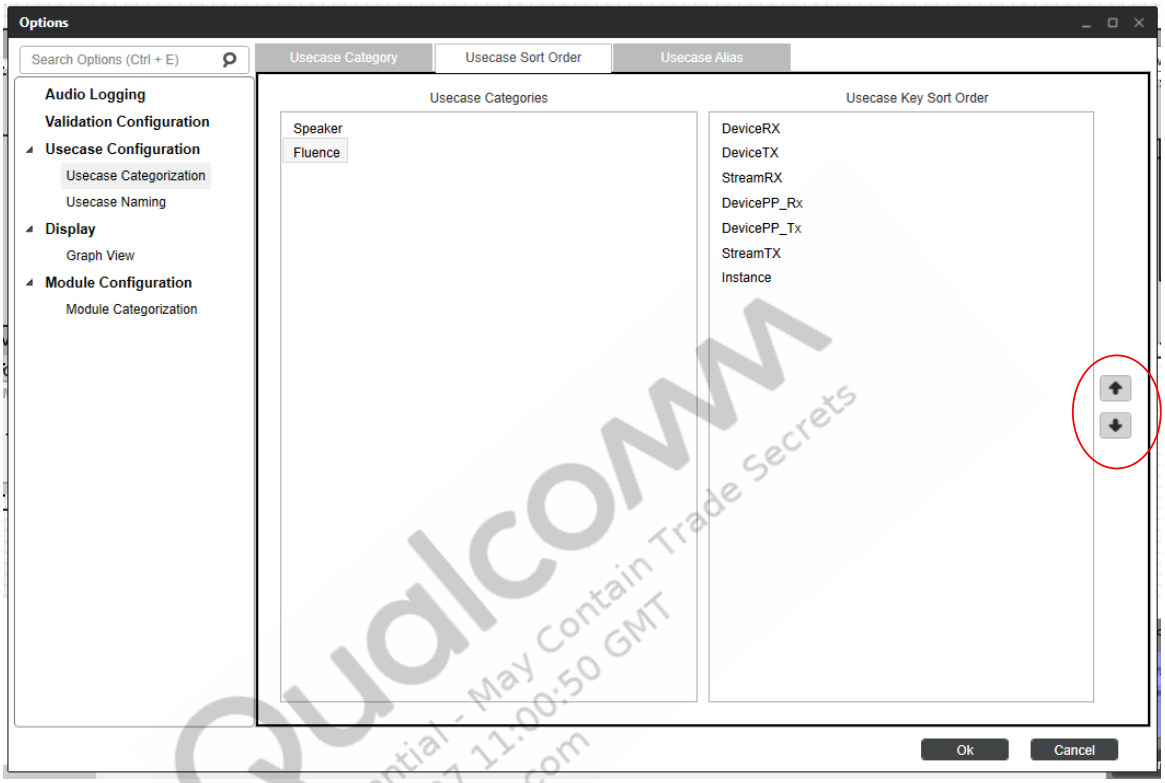
- Add a new Usecase Category – Select use cases from the left view and click **+** to create a new category. The category name can be changed by double clicking on the new category.
- Remove a Usecase Category – Select the category and click **X**.
- Add use cases to an existing Usecase Category – Select an existing Usecase Category from the right side, then select use cases from left side, and click the → button.
- Deleting a use case from a Usecase Category – Select the use case under the Usecase Category, and click **X**.
- Search for a use case – Use the top search bar to search for a specific term (for example, “Fluence") to reduce the use case list to only matching entries.



4.6.1.2 Use case sort order

Use case sort order can be set in the second tab for Usecase Categorization. Sort order allows to change the key (for example, StreamTX) position in the use case name for all use cases for a particular use case category.

The user can change the sort order by clicking a Usecase Category on the left side, selecting a key on right, and then clicking on either the UP or DOWN arrow to move its position in the Usecase Key Sort Order list.



4.6.1.3 Use case alias

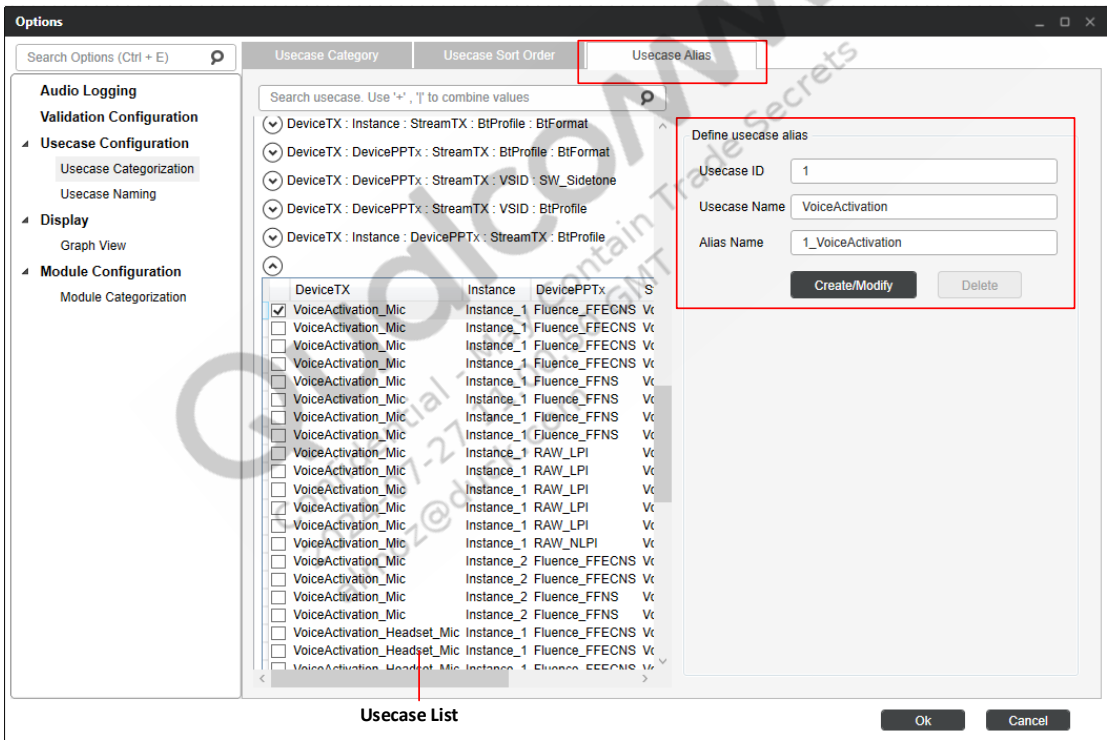
The user can set an alias for each use case instance for easy identification. This is particularly useful when use case names are long.

Use case alias can be set in the third tab for Usecase Categorization.

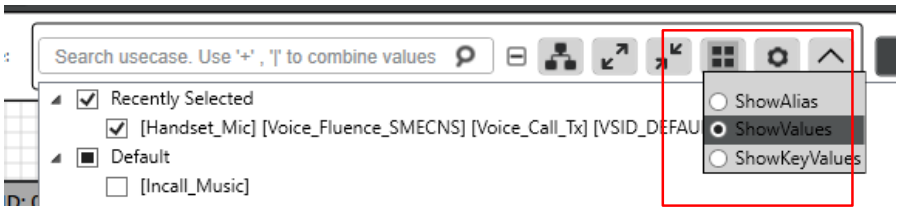
- Add a Usecase Alias – Select a use case from the use case list on the right. This will automatically assign a default Usecase ID and Usecase Name for the use case, both of which can be edited by the user. Click **Create/Modify** to create the use case alias.

  The Alias Name is a concatenated string made up of the Usecase ID and Usecase Name. The Usecase IDs are unique in an ACDB file.

- Remove a Usecase Alias – Select the use case from the alias list on the left (this is on the top of the use case list) and click **Delete**.



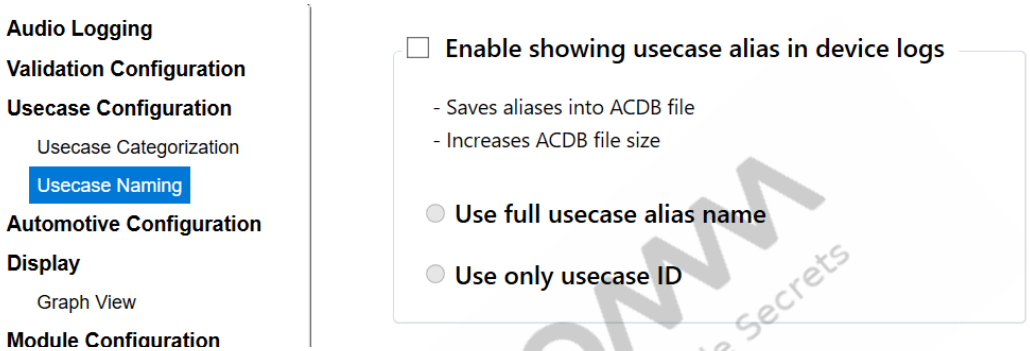
Once the alias is set, the user can use the option on the Usecase dropdown to show aliases, instead of Values or KeyValues, for the use cases which have a use case configured.



4.6.1.4 Use case alias in ACDB

A use case alias can be saved in a ACDB file and be used in logs instead of a key:value.

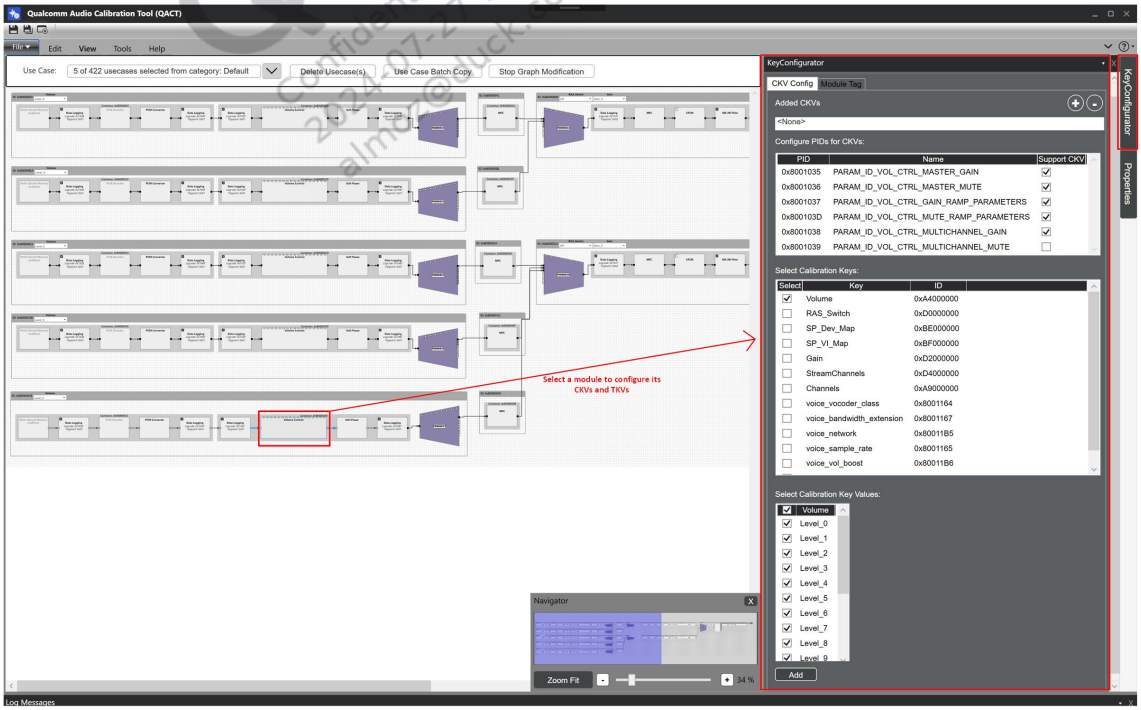
After use case aliases are set for use cases, the user needs to enable the settings in **Options -> Usecase Configuration -> Usecase Naming** as shown in the following image. The user also has the option to use full a use case alias name or just use case IDs.



4.7 Associate calibration IDs (CKV)

CKVs allow module instances to have different calibration data dependency. To configure CKVs, select a module and add or delete its CKV in the **KeyConfigurator -> CKV Config** tab.

The following diagram shows all capabilities of Key Configurator for CKV configuration.

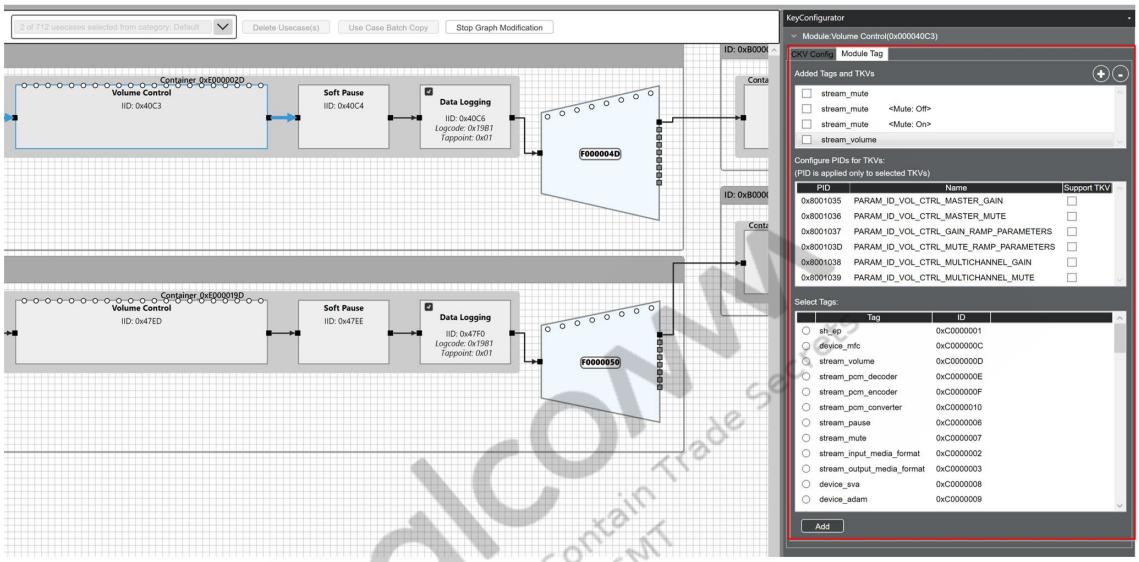


In the Configure PIDs for CKVs section, check or uncheck to add or remove calibration data for a PID for all CKVs.

4.8 Associate module tags (TKV)

Module tags and TKVs are for module instances. To configure TKVs, select a module and add or delete its TKV in the **KeyConfigurator -> TKV Config** tab.

The following diagram shows all capabilities of Key Configurator for TKV configuration.

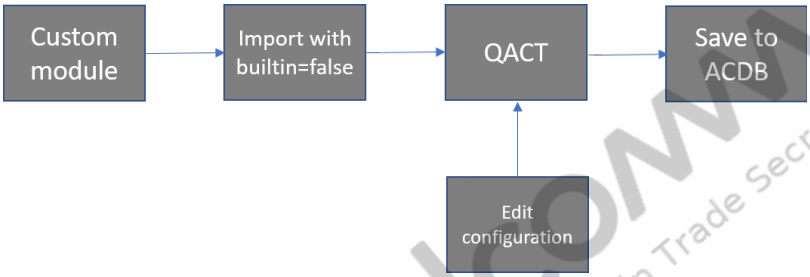


In the Configure PIDs for TKVs section, check or uncheck to determine if a PID should add or remove calibration data for selected TKVs.

4.9 Configure dynamic loading (AMDB)

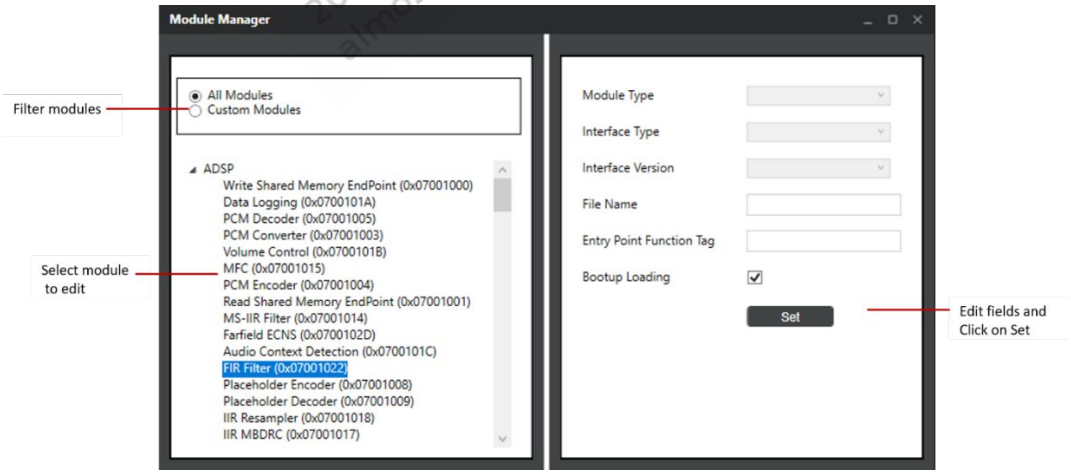
QACT provides an optional DSP module management function. The following figure shows the DSP module management workflow.

By default, if the module definition XML file does not contain the “builtin” attribute, it is considered a built-in module. Once module definitions are imported using Discovery Wizard, the user can modify when modules are loaded in the DSP build (see Section 4.9.1), potentially saving run-time memory and decreasing data processing time. Custom modules can also be registered and added to the DSP build (see Section 4.9.2). Upon saving, the ACDB file is updated to store the registration information for built-in and custom modules. It will be read by the DSP at the next bootup.



4.9.1 Edit built-in module configuration

1. Click **Tools -> Module Manager**.
2. Select a module.
3. Check/uncheck the Bootup Loading option accordingly.
4. Click **Set**.

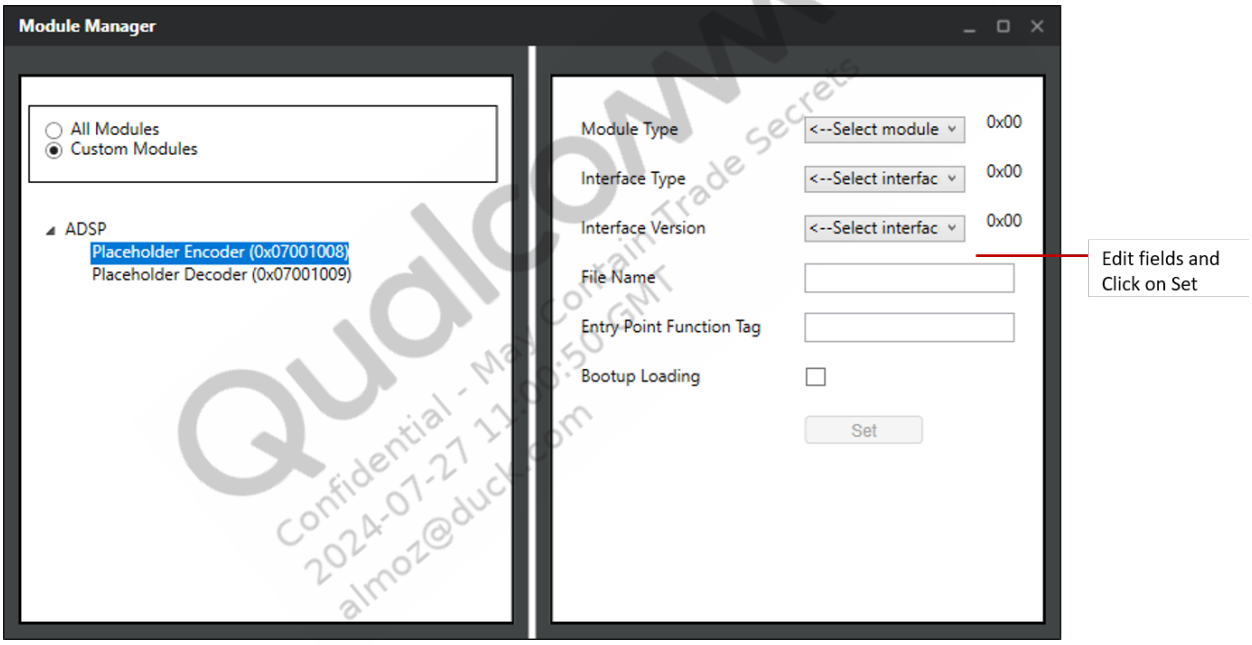


NOTE: For details on the Type and ID fields, see Section 4.9.2.

5. Click **File -> Save** or **File -> Save as** to save the ACDB files.
6. Push the file to /etc/acdbdata/ on the target. The file must be stored in this location in LA.

4.9.2 Register and modify custom modules

1. Import a custom module with “builtin=false” attribute using Discovery Wizard.
2. Click **Tools -> Module Manager**.
3. Select **Custom Modules** to filter the modules. The module added in Step 1 will be filtered.
4. Select the module.
5. Select the fields accordingly and click **Set**. See Section 4.9.2.1 for more information.
6. Click **File -> Save** or **File -> Save as** to save the ACDB files.
7. Push the file to /etc/acdbdata/ on the target. The file must be stored in this location.



All related custom .so files must be loaded on the target at the locations listed in the following table.

	Linux	Windows
ADSP_LIBRARY_PATH (path to where all .so files are pushed)	/system/vendor/lib/rfsa/adsp	c:\\Windows\System32\Qualcomm\RFSA\aDSP

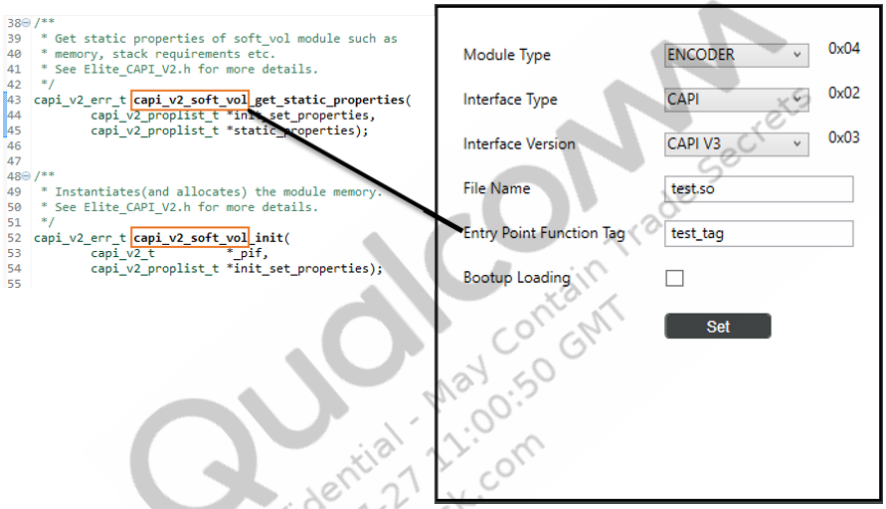
The following additional functions are available:

- To edit an existing module, click the module and modify the required fields
- To delete a module, delete the module from the ACDB files using Discovery Wizard

4.9.2.1 CAPI

The CAPI interface supports generic, encoder, decoder, converter, and packetizer module types. To add a CAPI module:

1. Select the module type accordingly.
2. Select the interface type as CAPI
3. Select the interface version as CAPI_V3.
4. Enter the .so file name.
5. In the Tag field, copy the tag that precedes the Get Static Properties and Init functions in the header (.h) file.



6. Check/uncheck the Bootup Loading field.
   - Unchecked – Module loads at the beginning of a specified use case
   - Checked – Module loads at bootup

4.10 Manage ACDB files using ACDB patch workflow

The ACDB patch workflow supports creation and application of ACDB patch files. ACDB patch files contain the delta between two ACDB files as a set of actions in readable format.

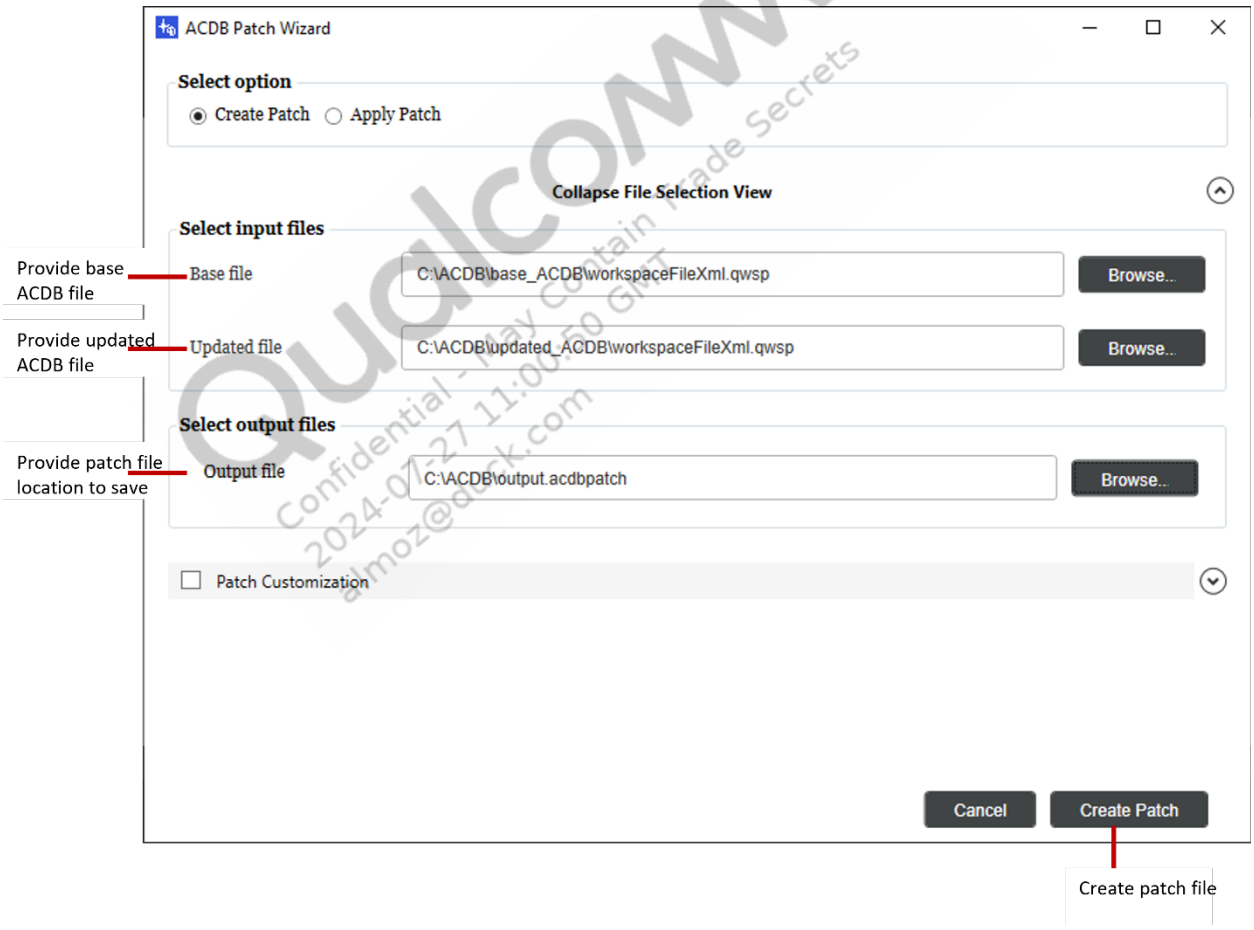
This section uses the following terminology:

- Updated ACDB – The ACDB file which has the latest and greatest calibration.
- Base ACDB – The ACDB file which has older calibration data.
- Target ACDB – The ACDB file which has customizations on top of Base ACDB.
- CKV – Calibration Key Vector. Used to define custom key/values for the calibration data associated to a module.
- TKV – Tag Key Vector. Used to define custom tags and corresponding key/values for the calibration data associated to a module.

4.10.1 Create ACDB Patch

To create an ACDB patch:

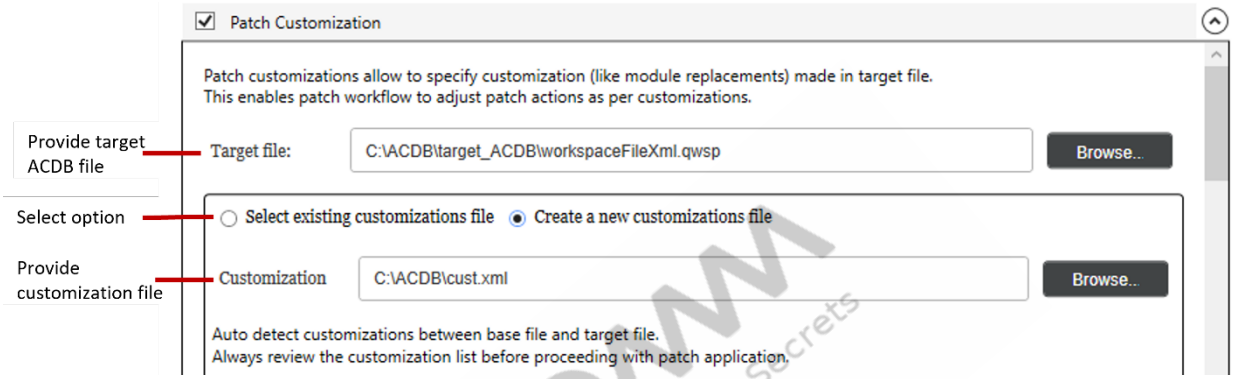
1. Click **Browse** next to “Base file” and select the Base ACDB file which has the older calibration.
2. Click **Browse** next to “Updated file” and select the Updated ACDB file which has the updated calibration.
3. Click **Browse** next “Output file” and provide the file path at which to save the patch file.
4. Enable **Patch Customizations** if applicable (see Section 4.10.1.1).
5. Click **Create Patch** to create the patch file.



Once **Create Patch** is clicked, the ACDB patch workflow will the compare base ACDB and updated ACDB to create a patch file. The patch file contains a delta of use cases, calibration data, properties, definitions, etc. between both the files as a set of actions which are in readable XML format. The patch file can be applied onto similar base files as explained in later sections. A summary of the create patch process is provided on the summary page.

4.10.1.1 Patch customizations

This feature enables understanding of customizations made by OEMs to QTI use cases such as module replacement, module removal, etc. during patch file creation which helps later during the application stage. This workflow takes a target ACDB as input. The target ACDB should be similar to the base ACDB file with additional customizations.



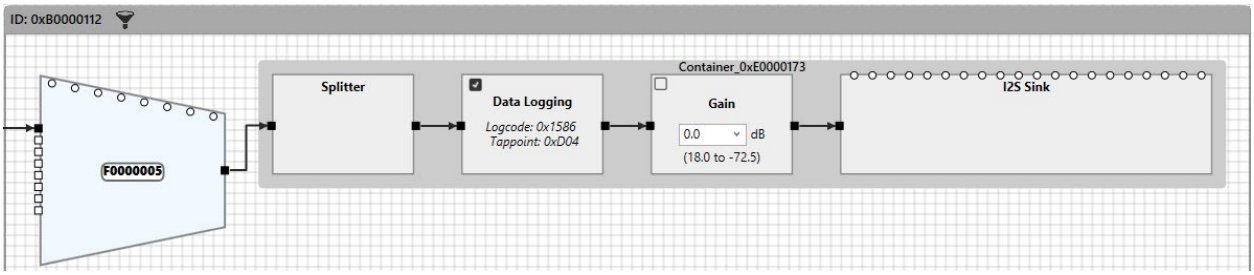
To use the patch customizations feature:

1. Click the **Patch customizations** checkbox to enable the patch customizations UX.
2. Click on **Browse** next to “Target file” and select the target ACDB file.
3. Click **Select existing customization file** if a customization file is already created using this workflow. Otherwise click **Create a new customization file**.
4. Click **Browse** next to “Customization” and provide the path to open from/save to based on step 3.

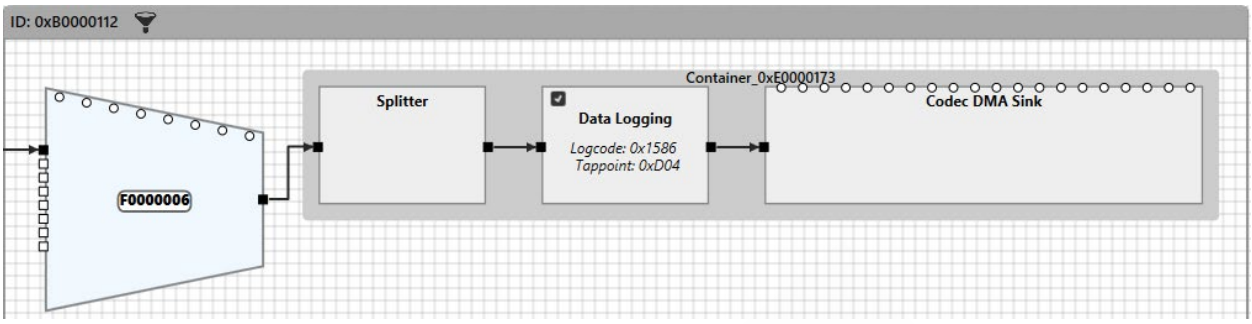
When **Create new customization file** is selected, the ACDB patch workflow will automatically detect customizations by comparing the base and target file. A summary of auto detection is provided for users to review and confirm. User review is required since there will be some scenarios where auto detection may be wrong.

For example, consider this following scenario for target and base subgraphs.

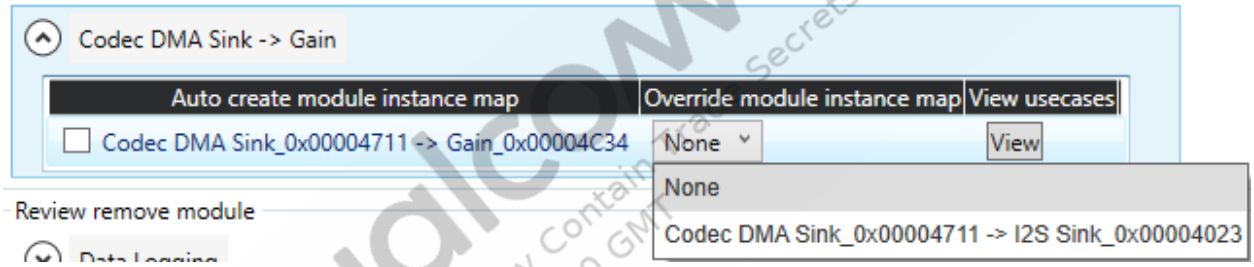
Target subgraph:



Base subgraph:



During comparison, the Gain module is mapped to Codec DMA Sink because of the module sequence, which is wrong. In the patch customizations, users will see the mapping as follows:



Users need to select the correct mapping to I2S Sink to correctly map this module instance.

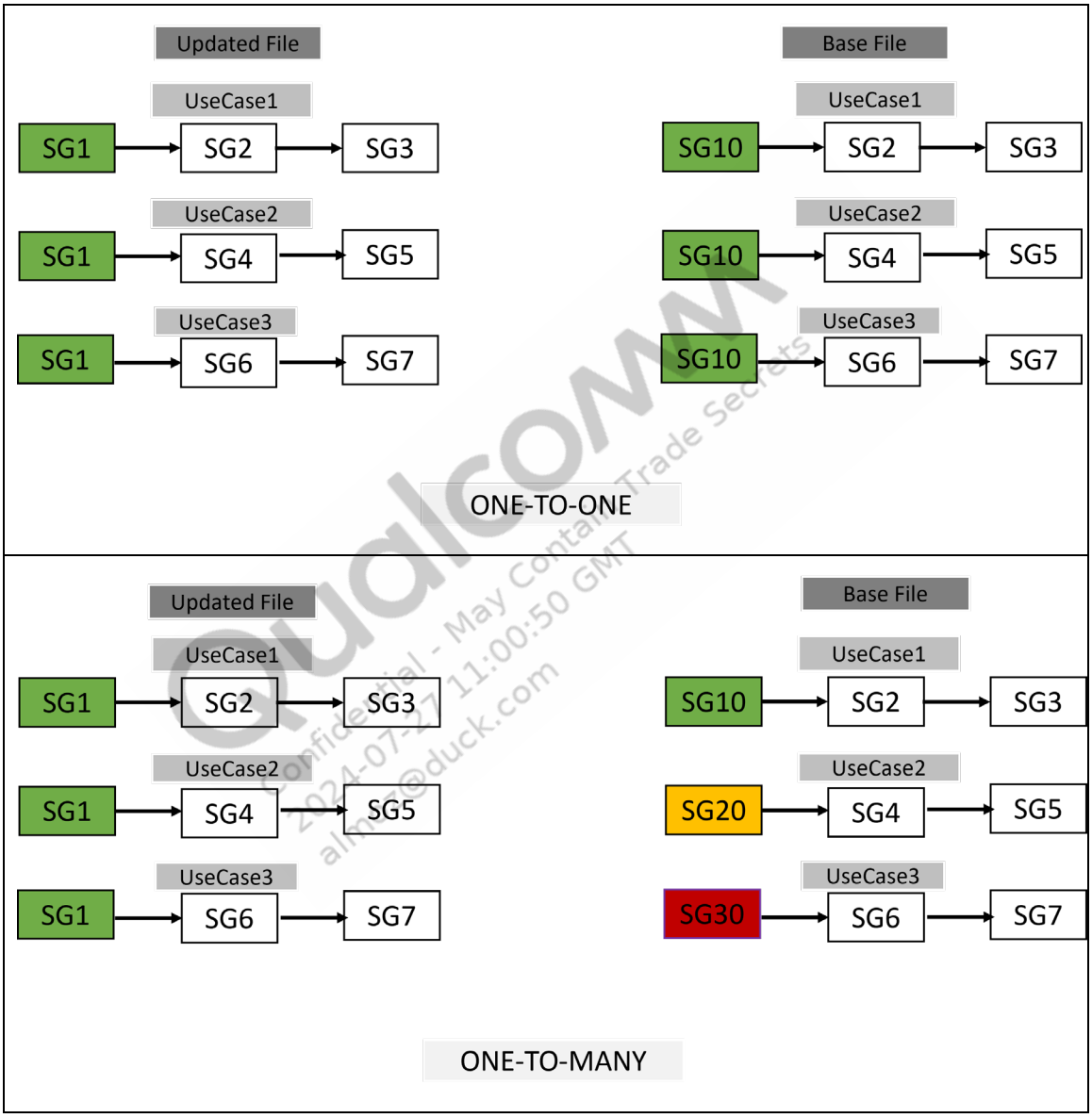
Currently, module replacement and module removal are supported as part of customizations. This customization information is later used during the application stage to modify actions accordingly.

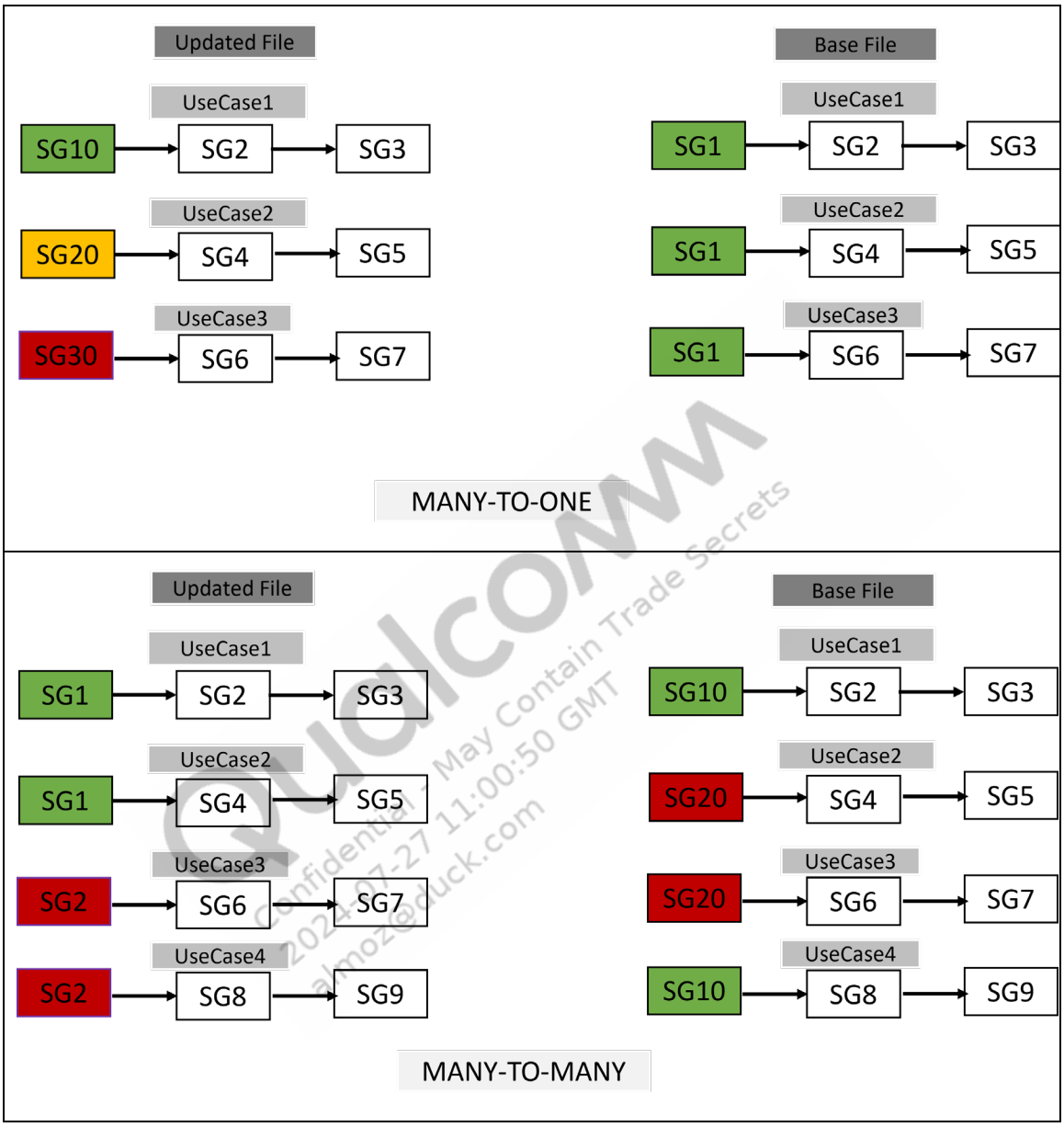
For example, If a control link needs to be added to the Codec DMA Sink module, but the target ACDB has I2S Sink in its place, using customization information, the control link is added to I2S Sink module directly.

4.10.1.2 Comparison concept

ACDB files can have the same use cases but subgraph/container/module IDs used can greatly vary between two files. Instance IDs cannot be used for comparison since they also can greatly vary. The comparison logic uses the subgraph sequencing information for common use cases to get a map of subgraphs. Once subgraphs are mapped, similarly modules are mapped based on the module sequence. Hence, the ACDB Patch and Diff/Merge features only work on use cases which are end-to-end connected.

Since the subgraphs used between the files can vary, there are possibilities to get ONE-TO-ONE, ONE-TO-MANY, MANY-TO-ONE and MANY-TO-MANY cases. These cases are explained in the following the images.



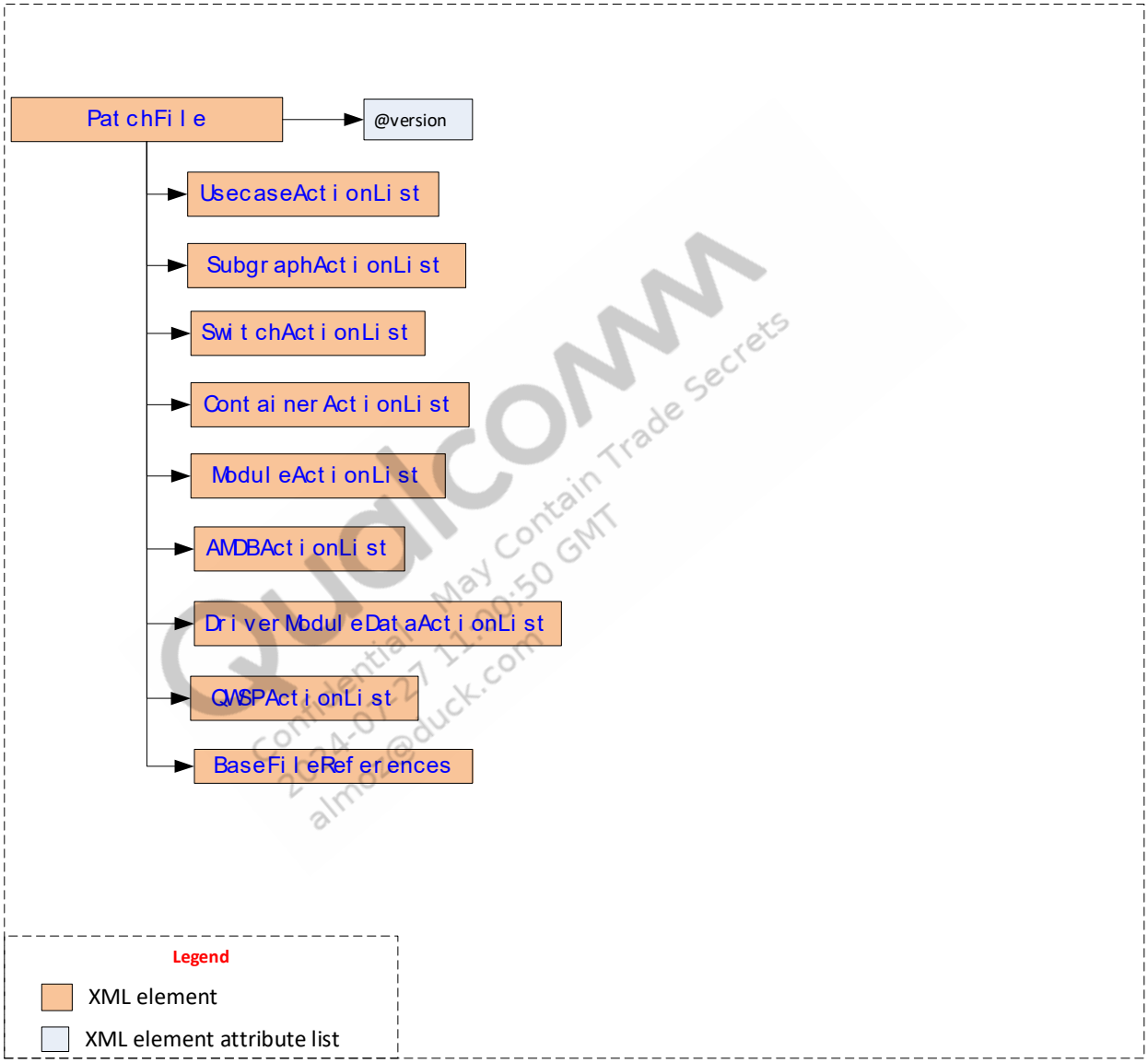


Map type	Actions during comparison
ONE-TO-ONE	Only one base file subgraph is mapped for the updated file subgraph.
ONE-TO_MANY	Multiple base file subgraphs are mapped for one updated file subgraph. Map the base file subgraph which has the highest number of matches in updated file.
MANY-TO-ONE	One base file subgraph is mapped to multiple updated subgraph. Map the base subgraph which has the highest number of matches in updated.
MANY-TO-MANY	Multiple base subgraphs are mapped for one updated subgraph and vice versa. Map the base subgraph which has the highest number of matches in updated.

Once a map is created, the information will be used while creating a patch, showing differences in diff/merge, etc.

4.10.1.3 ACDB patch file format

At a high level, patch files group actions as follows. Details of each group is provided in the following table. Complete details of supported actions are provided in Section 4.10.1.4.



Group	Comments
UsecaseActionList	Actions associated to use cases such as add, remove, etc.
SubgraphActionList	Actions associated to subgraphs such as add, remove, etc.
SwitchActionList	Actions associated to switches such as add, remove, etc.
ContainerActionList	Actions associated to containers such as add, remove, etc.

Group	Comments
ModuleActionList	Actions associated to modules such as add module, data links, control links, etc. Further grouped per module instances with connections under source module instance.
AMDBActionList	Actions associated to custom modules such as add, remove, etc.
DriverModuleDataActionList	Actions associated to driver modules such as add, remove, etc.
QWSPActionList	Actions associated to definitions like graph key, module definitions, etc.
BaseFileReferences	Information associated with base files. Used to convert labels to IDs during apply patch.

4.10.1.4 Supported actions in ACDB patch file

ACDB patch supports the following actions and associated sub actions.

Use case actions

Action	Sub action	Comments
add_usecase	add_sg, add_sw	Adds a new use case.
update_usecase	add_sg, add_sw, remove_sg, remove_sw	Updates an existing use case.
remove_usecase		Removes an existing use case.

Subgraph actions

Action	Sub action	Comments
add_sg	add_property	Adds a new subgraph.
update_sg	add_property update_property remove_property	Updates properties of an existing subgraph.
remove_sg		Removes an existing subgraph from ACDB file.

Switch actions

Action	Sub action	Comments
add_sw		Adds a new switch.
remove_sw		Removes an existing switch from ACDB file.

Container actions

Action	Sub action	Comments
add_cntr	add_property	Adds a new container.

Action	Sub action	Comments
update_cntr	add_property update_property remove_property	Updates properties of an existing container.
remove_cntr		Removes an existing container

Module actions

Action	Sub action	Comments
add_mod	add_cal_data add_tag_data add_module_tag	Adds a new module instance.
update_mod	add_cal_data update_cal_data remove_cal_data add_tag_data update_tag_data remove_tag_data add_module_tag remove_module_tag	Updates calibration of existing module instance.
remove_mod		Removes an existing module instance
add_dataconn	add_metadataconn	Adds a new data link
update_dataconn	add_metadataconn remove_metadataconn	Updates path of an existing data link
remove_dataconn	remove_metadataconn	Removes an existing data link.
add_ctrlconn	add_metactrlconn set_prop_data	Adds a new control link
update_ctrlconn	add_metactrlconn remove_metactrlconn set_prop_data	Updates path of an existing control link
remove_ctrlconn	remove_metactrlconn	Removes an existing control link.

Custom module actions (AMDB)

Action	Sub action	Comments
add_custom_module		Adds custom module information
update_custom_module		Updates custom module information
remove_custom_module		Removes custom module information

Driver module actions

Action	Sub action	Comments
add_driver_module	add_cal_data	Adds a new driver module
update_driver_module	add_cal_data update_cal_data remove_cal_data	Updates calibration of existing driver module
remove_driver_module		Removes existing driver module

Workspace actions

Action	Sub action	Comments
add_usecase_category	add_usecase	Adds a new use case category
update_usecase_category	add_usecase remove_usecase	Updates existing use case category
remove_usecase_category		Removes existing use case category
add_usecase_alias		Adds a new use case alias
update_usecase_alias		Updates existing use case alias
remove_usecase_alias		Removes existing use case alias
add_mod_def	addParam	Adds a new module definition
update_mod_def	addParam updateParam removeParam	Updates existing module definition
remove_mod_def		Removes existing module definition
add_graph_key	addKey	Adds a new graph key
update_graph_key	addKey removeKey updateKey	Updates existing graph key
remove_graph_key		Removes existing graph key
add_cal_key	addKey	Adds new cal key
update_cal_key	addKey removeKey updateKey	Updates existing cal key
remove_cal_key		Removes existing cal key
add_tag	add_tag_key	Adds new tag
update_tag	add_tag_key remove_tag_key update_tag_key	Updates existing tag
remove_tag		Removes existing tag
add_gecko_prop		Adds new subgraph/container property
update_gecko_prop		Updates existing subgraph/container property
remove_gecko_prop		Removes existing subgraph/container property
add_gsl_prop		Adds new GSL property
update_gsl_prop		Updates existing GSL property
remove_gsl_prop		Removes existing GSL property

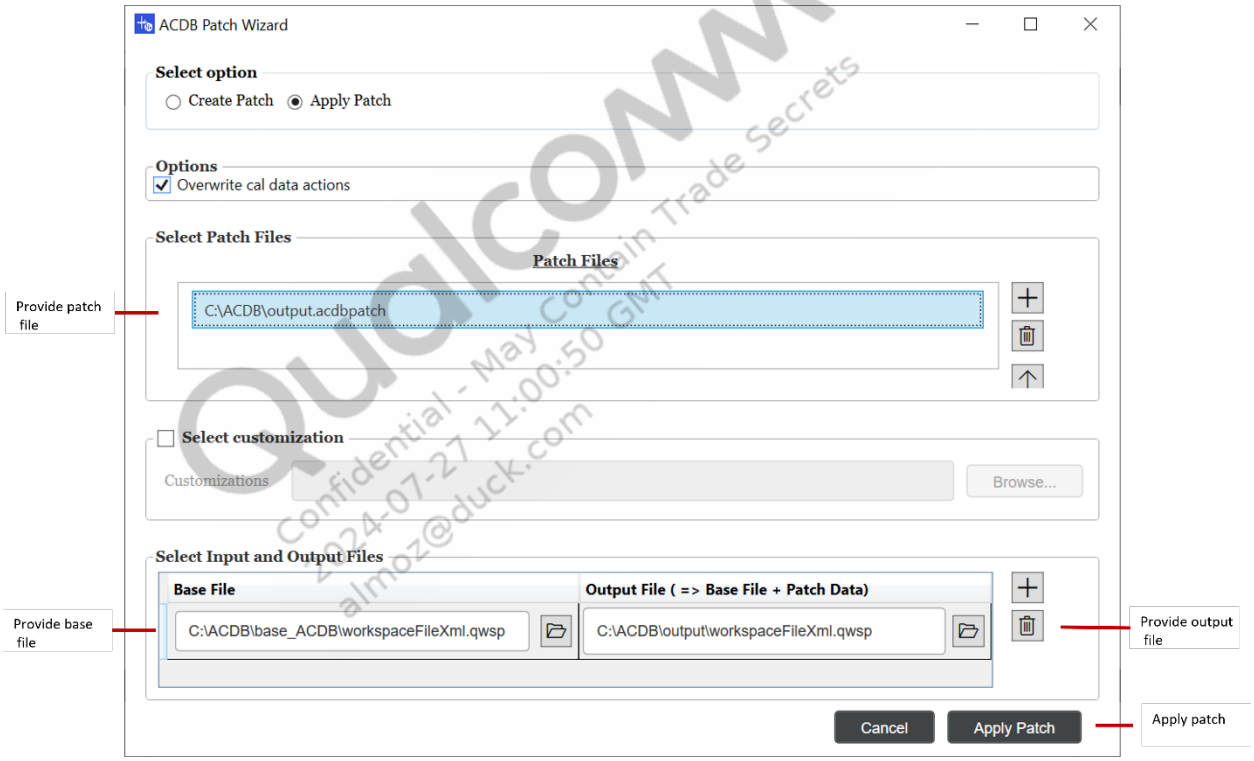
4.10.2 Apply ACDB Patch

A patch file can be applied onto an ACDB file with the set of associated actions. A patch file can be applied to multiple ACDB files by adding multiple base files in the UX.

To apply a patch file:

1. Select the **Apply Patch** option button.

2. Select **Overwrite cal data actions** if applicable. This option is not selected by default in order to not overwrite cal data (since the files can have calibration completed already).
3. Click **+** and select the patch file(s). Multiple patch files can be applied at once. However, the sequence of patch files is important, so ensure the sequence is selected correctly using the arrow buttons.
4. Click **Select customizations** if applicable and provide the customization XML file path.
5. Select the base file onto which the patch should be applied. The output file represents the file to be saved after patch application.
6. Click **Apply Patch** to apply the patch file(s).



A summary of patch file application is provided. In case any error occurs during the process, the process is aborted and error message with details is provided.

Any warnings/invalid actions during application of patch are provided at the end of the process. Users should go through the warnings/invalid actions and confirm them before using the generated output file. Details of errors/warnings can be exported by clicking **Export Errors Details**.

4.10.3 Recommended workflow for using ACDB Patch

As mentioned in the previous sections, the ACDB patch workflow works well for similar ACDB files. Use cases should be connected end-to-end and have the same switch placements for creation/application of patch files to work as expected.

QTI recommends ensuring the same switch placements are present in use cases before creating/applying an ACDB patch. In case of different switch placements between ACDB files, users should review the generated output file before using it.

The following is the recommended workflow for ACDB patches.

1. Create patch by providing base and updated files.
   - In case of customization, create customization xml by providing the target file.
2. Apply the patch file using customization (if applicable) onto the base file, generating the output file.
3. Analyze the warnings/invalid actions provided in the summary page at the end of apply patch workflow.
   - For warnings, manually fix the output file for those use cases either by graph designer or diff/merge (whichever is appropriate)

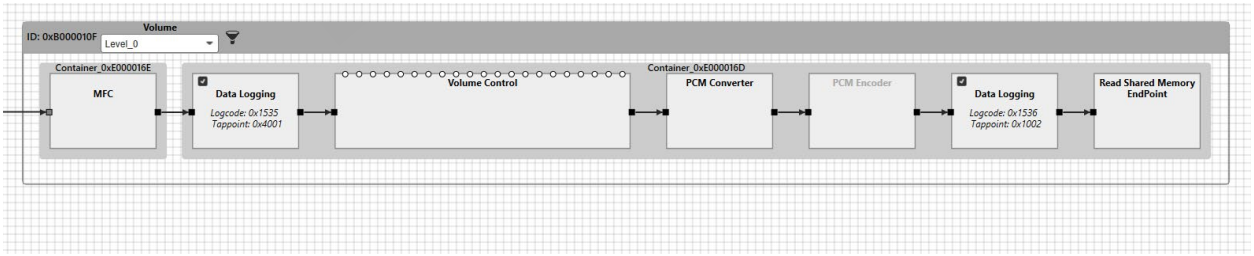
4.10.4 Limitations of using ACDB patches

4.10.4.1 Scenario 1: Different switch placements in base and target files

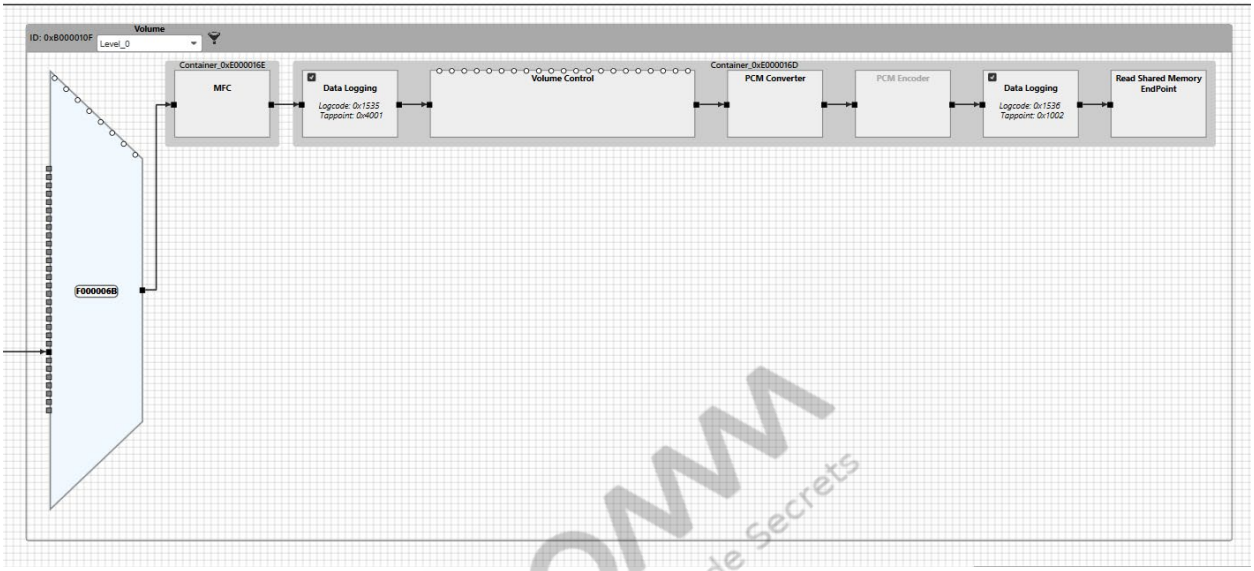
Scenario: New use case is added using a shared subgraph.

Subgraph description

Base file and updated file have a subgraph as follows:



Target file has a subgraph as follows:

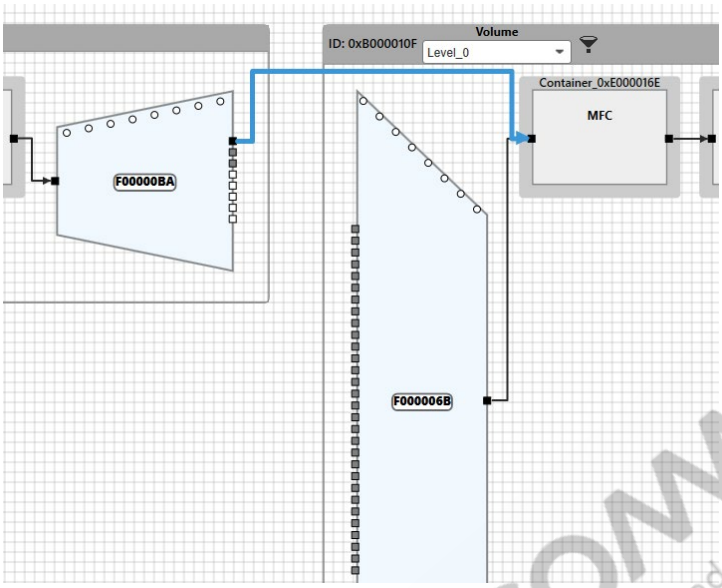


There is a switch in target file which is not present in base/updated files.

Patch action

```
<add_dataconn srciidLabel="UPDATED_Splitter_0x000042A1"
srcportId="0x00000001" destiidLabel="BASE_MFC_0x000046F9"
destportId="0x00000002">
        <SubActions>
          <add_metadataconn type="MODULE_TO_SWITCH"
srcLabel="UPDATED_Splitter_0x000042A1" srcportId="0x00000001"
destLabel="UPDATEDSW_0xF0000064" internalSwConn="True"
destKv="[Instance:Instance_1,StreamTX:PCM_Record]" />
          <add_metadataconn type="SWITCH_TO_MODULE"
srcLabel="UPDATEDSW_0xF0000064" destLabel="BASE_MFC_0x000046F9"
destportId="0x00000002" srcKv="[Instance:Instance_1,StreamTX:PCM_Record]"
/>
        </SubActions>
 </add_dataconn>
```

Output file after patch action:



Since patch action is not aware of the switches, the connection is added directly to the MFC module.

Recommended workaround:

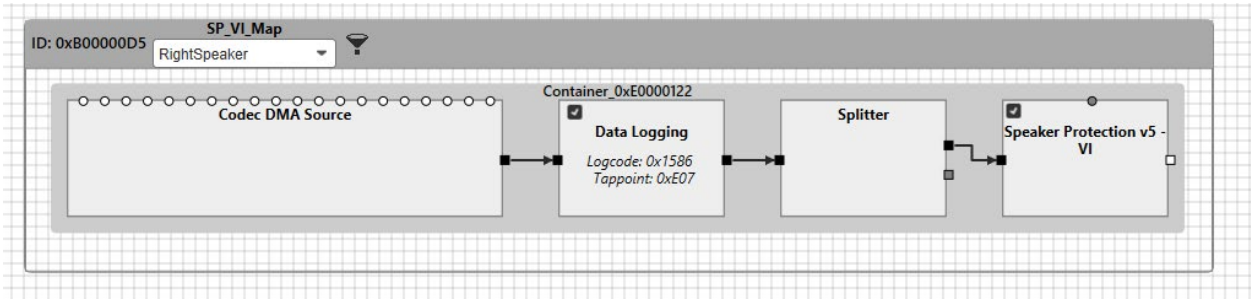
After applying the patch, correct the connection in output file manually using graph designer and correct the routing.

4.10.4.2 Scenario 2: Different switch placements in base and target files

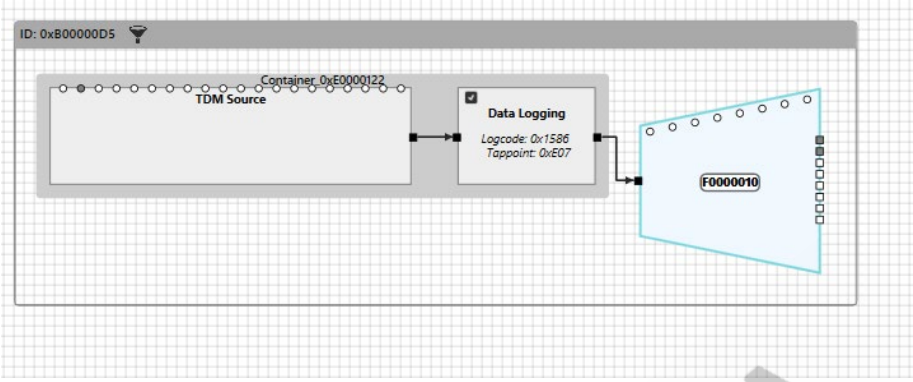
Scenario: New use case is added using a shared subgraph, but addition of subgraph connection failed.

Subgraph description

Base file and update filed have a subgraph as follows.



Target file has a subgraph as follows.

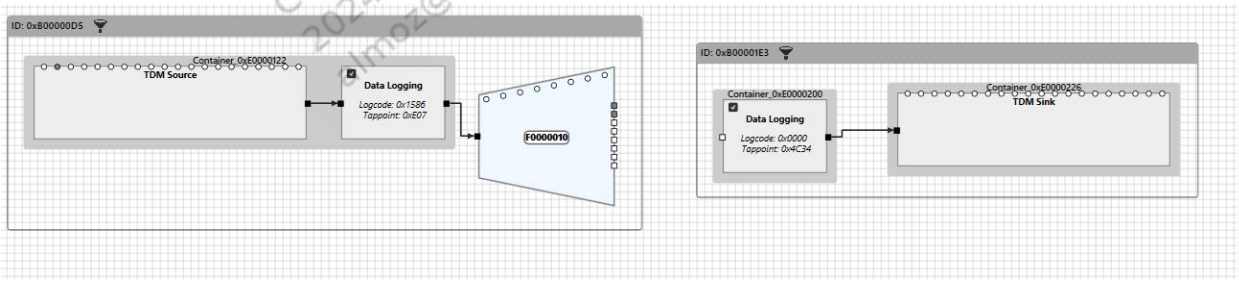


There is a switch in target file which is not present in base/updated files. Update file uses splitter module for the new subgraph connection for the new use case.

Patch action:

```
<add_dataconn srciidLabel="Base_Splitter_0x000042A1" srcportId="0x00000003"
destiidLabel="BASE_DataLogging_0x000046F9" destportId="0x00000002">
        <SubActions>
        </SubActions>
 </add_dataconn>
```

Output file after patch action:



The connection is not added as there is no splitter module corresponding to the base file in the target file. Instead, a switch is present, so the source of connection is missing. This is thrown as a validation error in path as an invalid action. Also, a graph routing error (E141)) will be shown as part of validation

Recommended workaround

After applying the patch, correct the connection in the output file manually using graph designer and correct the routing.

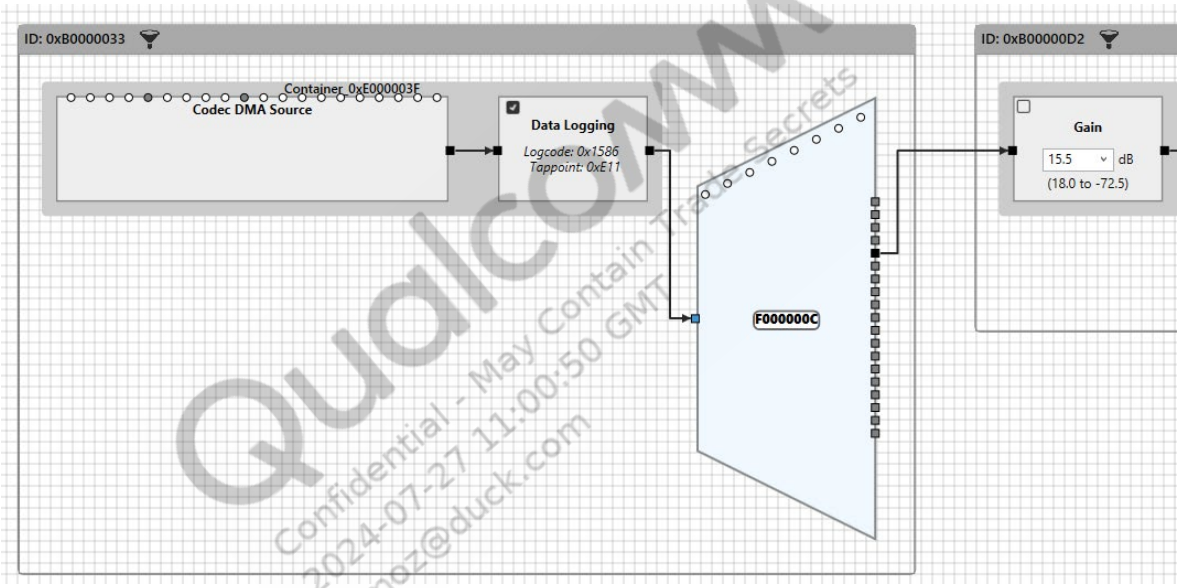
4.10.4.3 Scenario 3: Different switch placements in base and target files

Scenario: New use case is added using a shared subgraph, but addition of KVs to switch ports failed.

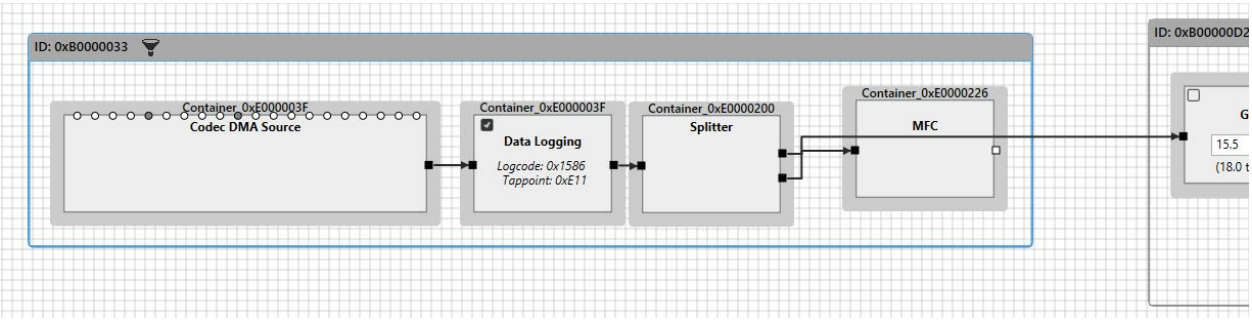
Subgraph description

Base file and updated file have a subgraph as shown below. The Speaker_Mic +Instance_1+PCM_Record use case below is present in both the base and updated files, but the output port KVs in both files are different.

- Base File output port KV : Speaker_Mic
- Updated File output port KV: Speaker_Mic, Headphone_Mic



Target file has a subgraph as follows.



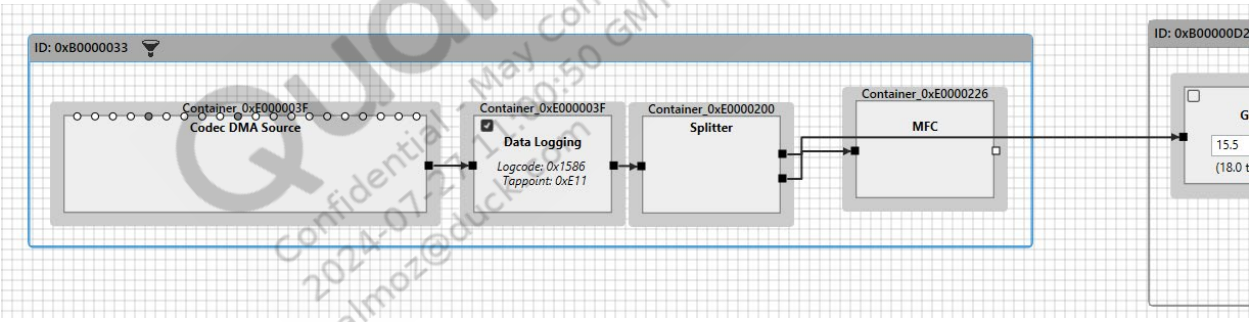
There is no switch in target file corresponding to 0xF000000C in the base and updated files.

Patch action

Add the new use case Headphone_Mic+Instance_1+PCM_Record and add the subgraph connection that will update the target port KV.

```
<add_dataconn srciidLabel="Base_Splitter_0x000042A1" srcportId="0x00000001"
destiidLabel="BASE_Gain_0x000046F9" destportId="0x00000002">
        <SubActions>
          <add_metadataconn type="MODULE_TO_SWITCH"
srcLabel="UPDATED_Splitter_0x000042A1" srcportId="0x00000001"
destLabel="UPDATEDSW_0xF000000C" internalSwConn="True"
destKv="[DeviceRX:Headphone_Mic]" />
          <add_metadataconn type="SWITCH_TO_MODULE"
srcLabel="UPDATEDSW_0xF000000C" destLabel="BASE_Gain_0x000046F9"
destportId="0x00000002"
srcKv="[ DeviceRX:Headphone_Mic]" />
        </SubActions>
 </add_dataconn>
```

Output file after patch action:



In this case, the use case will be added, but no KVs will be updated as there is no corresponding switch in the target file. A validation error is thrown mentioning that the action is invalid (connection addition failed as source is missing). Also, a graph routing error (E141)) will be shown as part of validation.

Recommended workaround

After applying the patch, routing has to be corrected by adding a switch/editing the graph.

4.11 Manage ACDB files using Diff/Merge

The Diff/Merge Wizard supports the capability to add, delete, and update the supported use cases and definitions in ACDB files.

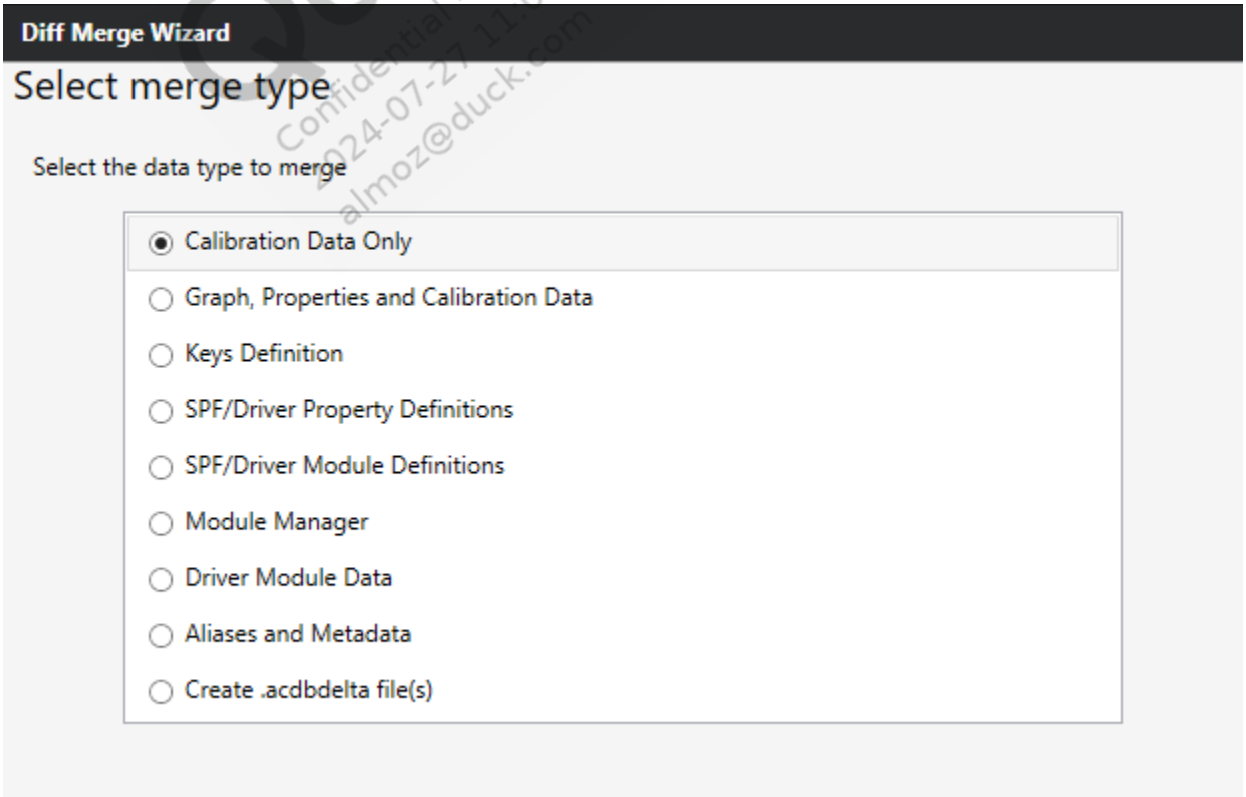
4.11.1 Terminology

This section lists the terminology in diff/merge that will be used in subsequent sections.

- **Reference ACDB** – The ACDB file which has the latest and greatest calibration. This file is never modified in diff/merge.
- **Target ACDB** – The ACDB file which needs to be modified during diff/merge to receive the changes.
- **CKV** – Calibration Key Vector. Used to define custom key/values for the calibration data associated to a module.
- **TKV** – Tag Key Vector. Used to define custom tags and corresponding key/values for the calibration data associated to a module.

4.11.2 Supported options in diff/merge

The wizard takes a reference workspace file and a target file as input and provides a summary of various types of comparison that can be done on the provided files. This section summarizes when to use each option provided.



- **Calibration Data Only** – Supports comparison and merging of all the calibration data (CKV, TKV, Tagged modules)
- **Graph, Properties and Calibration data** – Supports comparison and merging of all the use cases and its associated data (subgraphs, connections, properties, calibration data etc). Please note: in case the use cases need any additional definitions to be added during merging, diff/merge will automatically add the required definitions onto the target ACDB file.
- **Key Definitions** – Supports comparison and merging of graph keys, cal keys and tag key definitions.
- **SPF/Driver Property Definitions** – Supports comparison and merging of all the SPF property and driver property definitions.
- **SPF/Driver Module Definitions** – Supports comparison and merging of all the SPF module and driver module definitions.
- **Module Manager** - Supports comparison and merging of module bootup loading configuration and custom module configuration data.
- **Driver module data** – Supports comparison and merging of driver module data.
- **Aliases and Meta data** - Supports comparison and merging of use case/module aliases meta data associated.
- **Create .acdbdelta files** – Supports creation of .acdbdelta files which contains differences of calibration data between the reference and target ACDB files.

4.11.3 Manage supported use cases in ACDB files

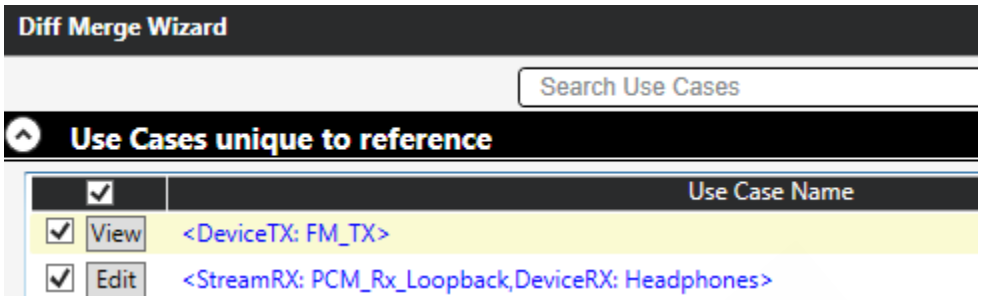
The wizard takes a reference workspace file and compares it with the target file to provide a summary of the use cases that can be added/deleted/updated in the ACDB files.

4.11.3.1 Add a new use case to ACDB

1. In the QACT main window, click **Diff/Merge ACDB files**.

2. Click **…** and select the target ACDB file.
3. Click **…** and select the reference ACDB file.
4. Click **OK**.
5. Select **Graph, Properties and Calibration Data**.
6. Click **Next**.

7. In the “Use cases unique to reference” section, select the use cases to add.



8. Click **Next**.
9. Click **Yes** to confirm to proceed.
10. Click **Finish** to save the target ACDB file.

NOTE: During merge operation, if any of the definitions (graph key, cal key, tag key, SPF property definition, module definition etc) associated to the new use case are not present in the target workspace file, diff/merge will automatically add the required definitions during the merge operation.

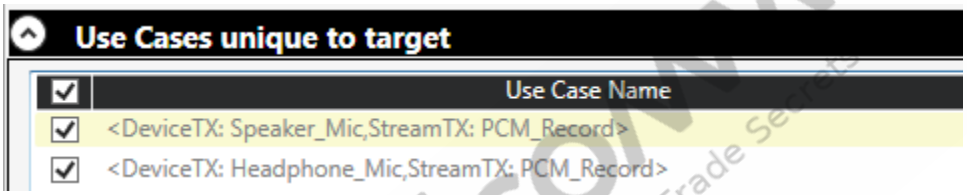
The selected use cases are added to the target workspace file and a summary of the updates is shown in the wizard.

The **View** button beside the checkbox denotes that the use case either uses all subgraphs which are newly added or shared subgraphs with no differences between reference and target or both.

The **Edit** button beside the checkbox denotes that one or more shared subgraphs have differences between the reference and target. Users can unselect the changes in shared subgraph to merge only the required changes onto the target file.

4.11.3.2 Delete a use case from ACDB

1. In the QACT main window, click **Diff/Merge ACDB files**.
2. Click **…** and select the target ACDB file.
3. Click **…** and select the reference ACDB file.
4. Click **Next**.
5. Select Graph, Properties and Calibration Data.
6. Click **Next**.
7. In the “GKVs unique to target” section, select the use cases to delete.
8. Click **Next**.



The selected use cases are deleted from the file and a summary of the updates are shown.

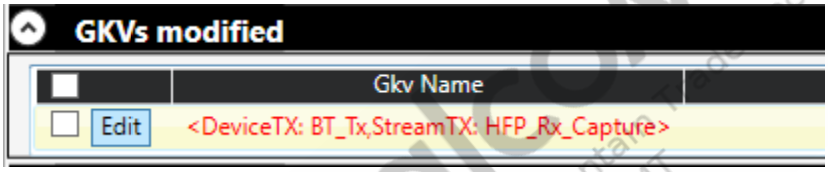
4.11.3.3 Modify a use case in ACDB

Modification includes the following changes:

- Add/delete a subgraph
- Add/delete an independent switch
- Add/delete a container inside subgraph
- Add/delete a module inside subgraph
- Add/delete a switch inside subgraph
- Add/delete data/control link
- Add/delete dangling data/control links
- Replace a module and associated connections
- Add/delete/modify subgraph/container properties
- Merge module data, calibration/tag data

To modify a use case:

1. In the QACT main window, click **Diff/Merge ACDB files**.
2. Click **…** and select the target ACDB file.
3. Click **…** and select the reference ACDB file.
4. Click **Next**.
5. Select **Graph, Properties and Calibration Data**.
6. Click **Next**.
7. In the “Modified Use cases” section, click **Edit** to show the differences in the use case. Differences between the target and reference are shown in the UI.
8. Select the required changes (more details below).
9. Click **Next.**

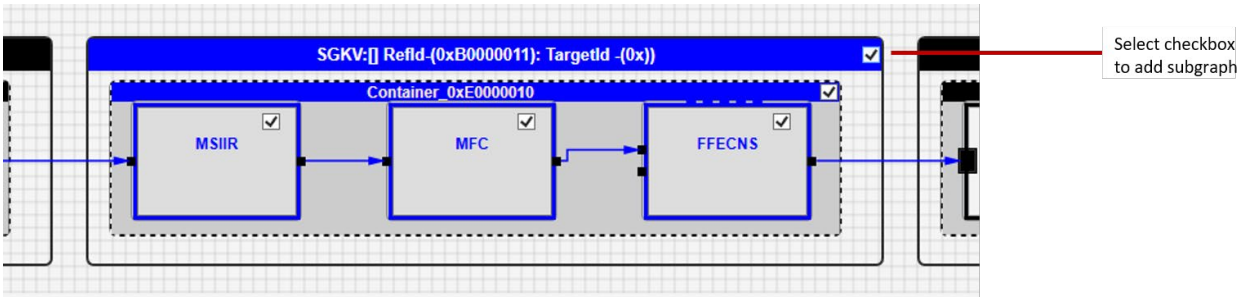


The use cases are modified accordingly, and a summary of the updates is shown.

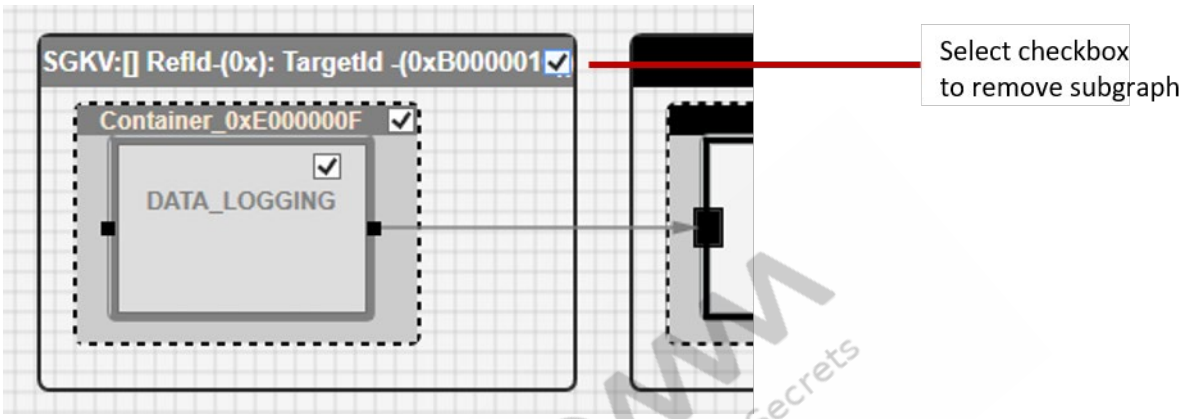
Add/delete a subgraph

During Edit, subgraph(s) which can be added from reference to target are marked in Blue. Select the checkbox on subgraph to add the subgraph to the target ACDB file. Once subgraph is selected, all the containers/modules are automatically selected. In addition to subgraph addition, the module/subgraph connections associated to the subgraph (marked in blue) will be added as well.

Note that during merge operation, if any of the definitions (cal key, tag key, SPF property definition, module definition etc.) associated to the new subgraph are not present in the target workspace file, diff/merge will automatically add the required definitions during the merge operation.

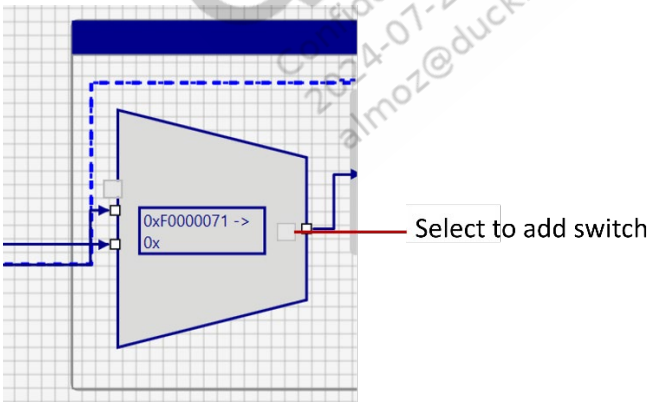


During Edit, subgraph(s) which can be removed in target file are marked in Gray. Select the checkbox on subgraph to remove the subgraph to the target ACDB file. Once subgraph is selected, all the containers/modules are automatically selected. In addition to subgraph deletion, the module/subgraph connections associated to the subgraph (marked in gray) will be deleted as well.



Add/delete a switch

During Edit, switch(es) that can be added from reference to target are marked in Blue. Select the checkbox on switch(es) to add the switch from the target ACDB file. In addition, the connections associated to the new switch (marked in blue) will be added and connections which are removed (marked in gray) will be removed.

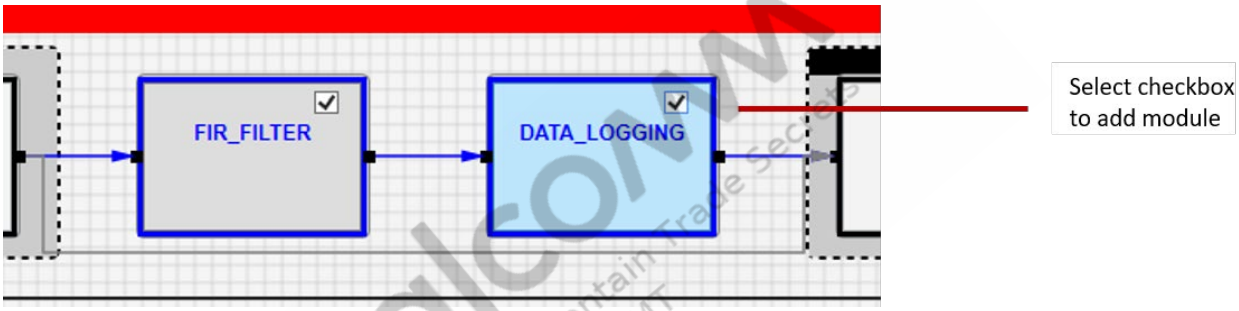


Not that adding/deleting a switch for an existing subgraph could lead to other side effects associated to graph modifications. Merge such use cases with caution.

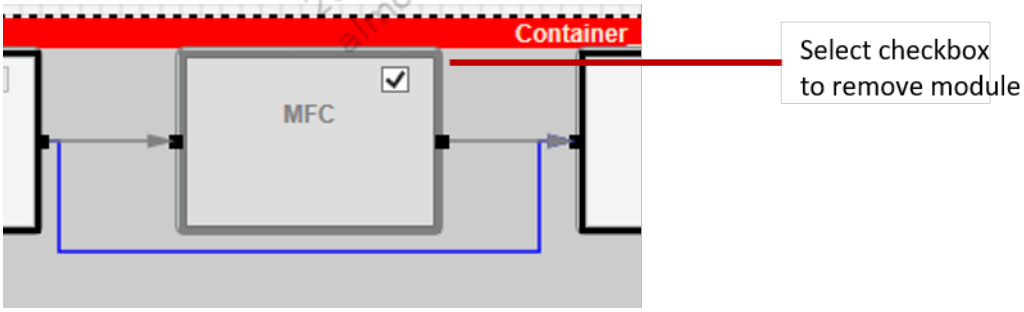
Add/delete a module

During Edit, module(s) which can be added from reference to target are marked in Blue. Select the checkbox on module(s) to add the module to the target ACDB file. In addition, the module/subgraph connections associated to the new module(s) (marked in blue) will be added and connections which are removed (marked in gray) will be removed as well. Also, the calibration/tag data associated with the module is merged from reference to the target ACDB file.

Note that during merge operation, if any of the definitions (cal key, tag key, module definition etc.) associated to the new module are not present in the target workspace file, diff/merge will automatically add the required definitions during the merge operation.



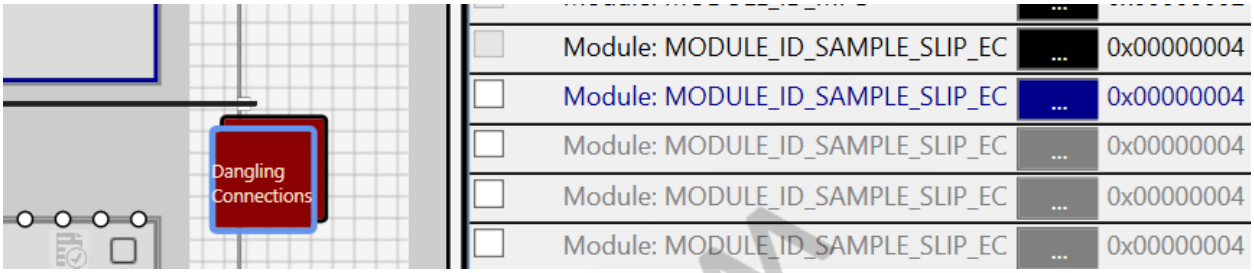
During Edit, module(s) which can be removed in target are marked in Gray. Select the checkbox on module(s) to remove the module from the target ACDB file. In addition, the module/subgraph connections associated to the removed module(s) (marked in gray) will be removed and connections which were added (marked in blue) will be added as well. Also, the calibration/tag data associated with the module is removed from the target ACDB file.



Note that, in case the modified subgraph is a shared subgraph across multiple use cases, selecting the modifications for merge will automatically select all the other use cases the subgraph is part of.

Add/delete dangling data/control links

Select the dangling link to display associated connections to add/delete. The connections which can be added are displayed in Blue and connections which can be removed are displayed in Gray. Black denotes common links between both the files.

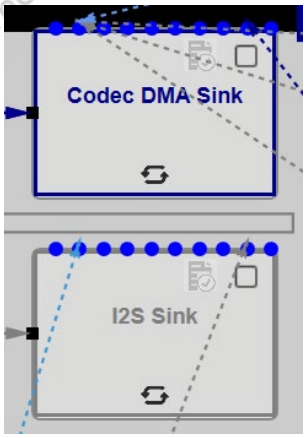


Replace a module and associated connections

Considering the cases where users made customizations to their files, there will arise cases where a module is replaced between reference and target file. In such cases, diff/merge provides capability to:

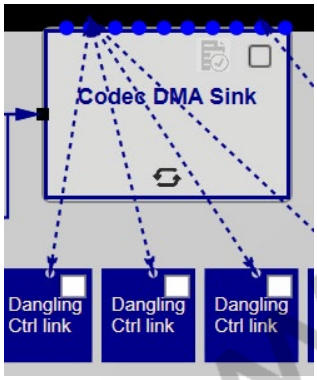
- Replace the target module with the reference module
  - Also support moving the existing connections to the target module to the reference module
- Keep target module and move the connections from reference module

The replaced modules are denoted by the replace icon as follows:

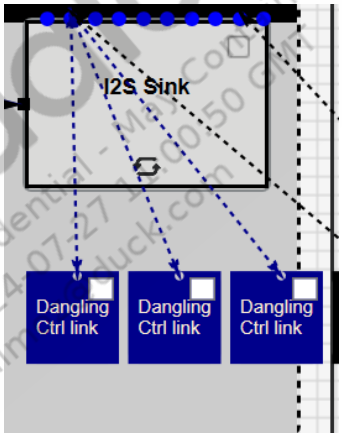


Users can then right click on the corresponding module to get options to replace.

- Right click on the reference module: Users can replace the target module with the reference module. Along with that, the connections associated to the target module can be added onto the reference module during merge.

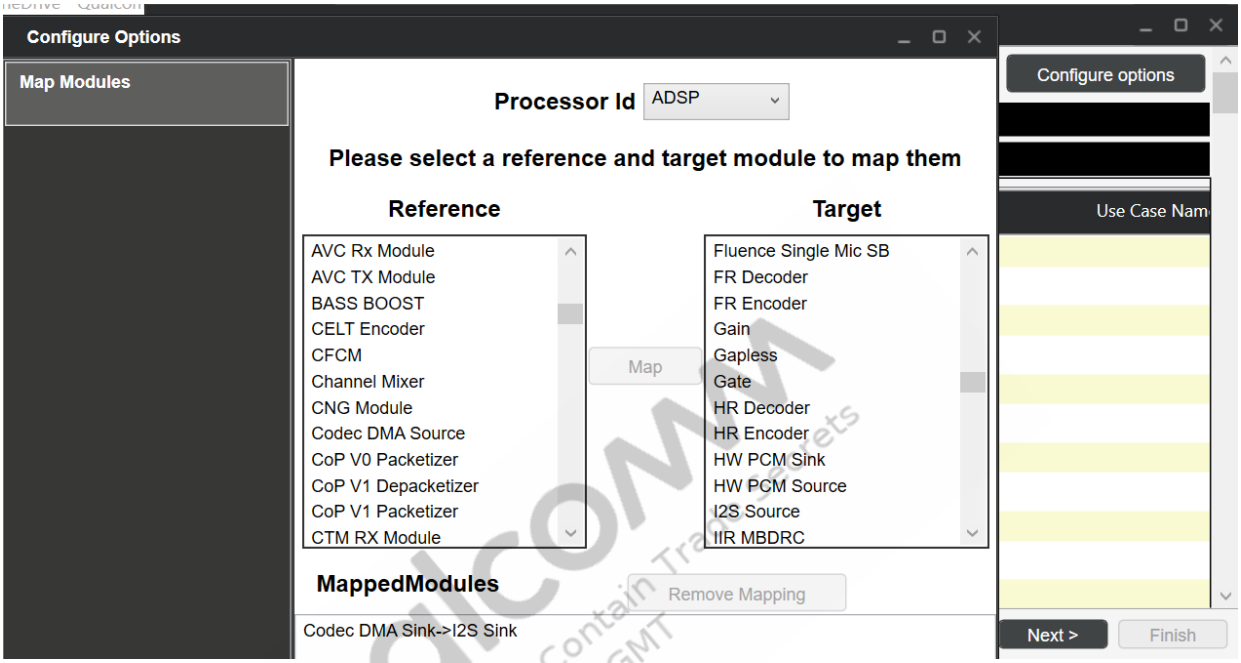


- Right click on the target module: Users can keep the target module and merge the connections associated to the reference module onto the target module. The selection is persisted and saved in the target workspace file. From next time forward, diff/merge will show the comparison with the selected option by default.



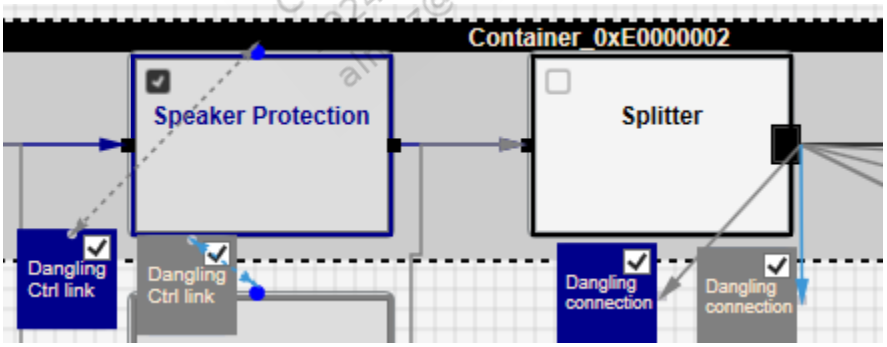
In case users changed their mind, they can click **Reset** to get back the original comparison view.

There can be some cases where the mapping is made to an adjacent module. In such cases, user can map modules using **Configure Options** and map those modules so that more weightage is given to the mapping.

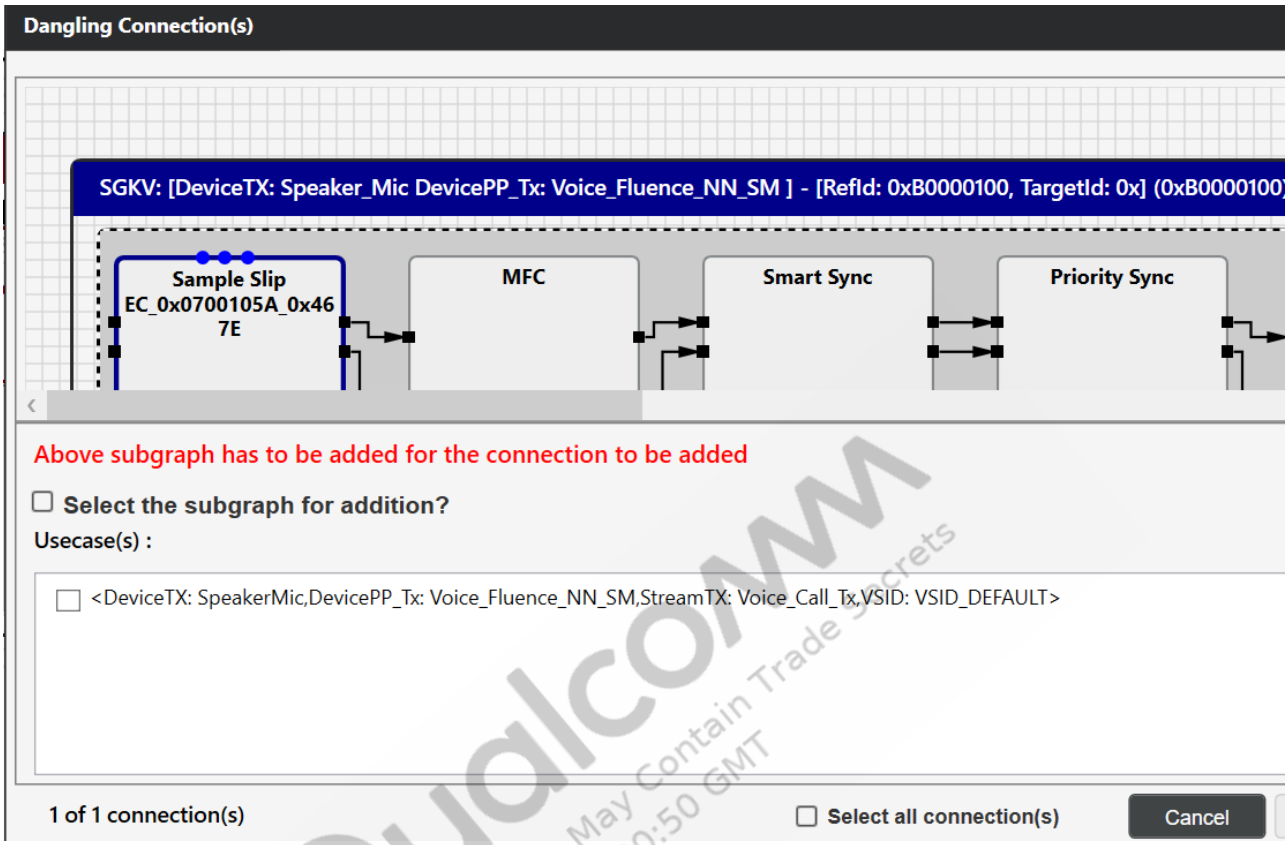


Add/delete a dangling data/control link

Dangling data/control links can be added/removed by selecting the checkbox on the control as shown below.



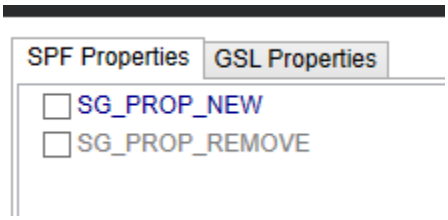
For a dangling data/control link to be added, the other end module/subgraph should also be marked for addition. In case the other end module/subgraph is not marked for addition, QACT shows a popup asking user for confirmation.



Add/delete/modify subgraph/container properties

Users can double click on a subgraph/container to view the differences in the properties.

Select the checkboxes so that the properties are added/deleted. Double click on a property to view the differences.



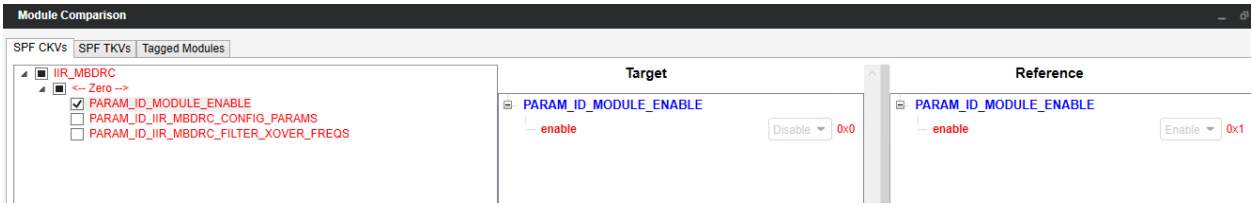
Note that during merge operation, if any of the definitions (SPF, driver properties) associated to the property are not present in the target workspace file, diff/merge will automatically add the required definitions during the merge operation.

Merge module data, calibration/tag data

During Edit, module(s) which are marked in Red indicate a difference in calibration/tag data change between both the files. Double click on the module to show the changes associated to the module.

The UX contains three tabs:

- SPF CKVs
- SPF TKVs
- Tagged modules



Changes in red indicate different data between reference and target. Blue indicates new data in reference which will be inserted to target file once selected and grey indicates data present in target which will be removed once selected. Once user selects the required changes, calibration data associated will be merged onto the target file.

Not that during merge operation, if any of the definitions (cal key, tag key) associated to the new calibration data are not present in the target workspace file, diff/merge will automatically add the required definitions during the merge operation.

Selecting the checkbox on module(s) in the main UX will merge all changes associated to the module (CKV, TKV and tagged module).

4.11.4 Merge files using review feature

Diff/merge supports the ability to mark items (use cases, subgraphs, containers, and modules) as reviewed so that users who merge files with several changes or across sessions can make a note of the items which are reviewed.

The following review icons are present for the items to indicate if an item is reviewed or not.

	Review is not done
	Review is done

Note: Once an item is marked as reviewed and the user clicks Finish at the end of merge process, the review selections are stored in the target workspace file and restored in the next session.

At start of merging process, the user needs to do the following steps:

1. Open Diff/Merge and select reference and target QWSP files.
2. For each changed use case:
   - Click **Edit** and inspect changes.

- For each change, if change needs to be merged, merge the change. Else, click the Review done icon.
- For each use case, click the Review done icon.

3. Click **Next** and **Finish** to merge/save current session updates.
4. From the next diff/merge onwards, saved reviews will be shown once diff/merge is opened again.

If the user marks reviewed at the use case level, all subgraphs/containers/modules will be marked as reviewed and are no longer editable. If the user needs to reset for a use case, they can click the Review done icon again.

If the user needs to reset all reviews click **Reset All Reviews**.

The following table explains what happens when an item is marked as review done.

Item is marked review done	What happens?
Use case	All subgraphs, containers, modules, subgraph data/control links are disabled for further selection.
Subgraph	All containers, modules, module and dangling data/control links, subgraph properties are disabled for further selection.
Container	Container and container properties are disabled for further selection
Module	Module, cal data, tag data, and tagged modules associated to a module is disabled for further selection.

Once a shared subgraph is marked as “Review done”, it is also marked done in other use cases. Once all subgraphs are marked as “Review done”, the use case is also marked as “Review done”.

4.12 Validation Manager

The Validation Manager framework is used to identify issues in ACDB files. QACT supports a list of validations which is run during graph operations and ACDB file save/save as.

The following are the types of validations:

- Active validations – QACT runs these validations during graph operation
- Passive validations – QACT runs these validations during ACDB file save/save as

4.12.1 Validation list

The following table lists the validations supported in QACT.

Validation name	Description	Possible errors
Validate SVA	▪ Validate all SVA use cases for SVA module ▪ Validate SVA module has a validate and connected path to the microphone ▪ Validate SVA Reference Use Cases are present	▪ E138: Missing SVA/ECNS module in SVA QC use case ▪ C101: SVA module is not connected to microphone ▪ C102: Missing SVA QC use case ▪ C103: SVA QC use case is modified incorrectly
Validate container properties	▪ Checks for missing container properties ▪ Checks for out-of-range container property data	▪ E107: Container property missing ▪ E108: Invalid data for container property ▪ W104: Invalid container stack size
Validate SPF properties	▪ Checks for missing subgraph properties ▪ Checks for out-of-range subgraph property data	▪ E113: Subgraph property missing ▪ E114: Invalid data for subgraph property
Validate GSL properties	▪ Checks for missing GSL properties ▪ Checks for out-of-range GSL property data	▪ E115: GSL property missing ▪ E122: Invalid driver property data error
Check for invalid connections	▪ Checks for duplicate data connections ▪ Checks for invalid data connections	▪ E100: Invalid data link error ▪ E101: Invalid control link ▪ E119: Duplicate data link error ▪ E120: Duplicate control link error ▪ E121: Invalid dangling link error
Module cal data validation	Checks for out-of-range calibration data for all modules	▪ E108: Invalid data for container property ▪ E109: Invalid calibration data ▪ E117: Container not found error ▪ E118: Param not found in module error ▪ E104: Module not found error
Validate multichannel module calibration data	Checks whether all params in multichannel modules have the same channels or not	E111: Invalid data for multichannel module
Validate module tool policy	Compares param tool policy and calibration data	E116: Tool policy and data not matching for module

Validation name	Description	Possible errors
Check for offload capability (active validation)	▪ Checks if offloading is valid or not ▪ Checks whether offloading modules are in sequence or not ▪ Checks for the presence of endpoint modules ▪ Checks for offloading modules in the offloading DSP	▪ E106: Invalid operation ▪ E103: Offloading non-sequence modules ▪ E104: Module not found error ▪ E105: Ports missing for module
Check for max module port property	Compares for maximum occupied port number with the one in module definition	E112: Occupied ports count is greater than max value in module
Validate All Graph Data	Validate all graph related data	▪ E130: some graph has conflict kvs. ▪ E131: some graphs have duplicated GKV ▪ W101: Some graph is not EndPoint to EndPoint ▪ W102: some graphs have ZERO GKV ▪ W103: Manually created graph has no subgraph pair connection
Validate MailBox module intent count	For any control link, if the source of destination module is mailbox, it should only have one intent assigned	▪ E126: Mailbox intent error
Validate meta graphs	Meta graph is internal data in QACT session and stored in workspace file. Should validate if a meta graph matches its graph data or not.	▪ E142: mismatch between meta graph and graph data
Check for module replace possibility	Check if a module can be replaced by another one or not	▪ E106: invalid operation
Validate GKVs by routing switch port KVs	Route switch port KVs of a graph to check routed GKV same as its GKV or not	▪ E141: Mismatch between a graph's GKV and routed GKV

Validation name	Description	Possible errors
Validate all Switch Data	Validate Switch data	▪ E132: Switch internal connection is missing. ▪ E133: Switch input and output ports have conflict kv ▪ E134: Switch has invalid internal connection for concurrency ▪ E135: Switch ctrl link has internal/external link missing or intent does not match ▪ E146: An independent switch is not used in any graph ▪ E147: Switch has no any input or output connection ▪ E148: Switch input/output port has more than one connections ▪ E149: Switch with internal module should not have internal connection from input port to output port directly

4.12.2 Error List

The following table lists the errors possible during validation and their auto-fix capabilities.

Error Code	Error string	Description
C101	SVAInvalidMicrophoneError	▪ Critical error: SVA module is not connected to microphone Solution: Refer to *Low Power Island Voice Activation – SVA 8.0* (80-40939-153) for resolution instructions.
C102	SVAMissingUseCaseError	▪ Critical error: Missing SVA QC Use case Solution: Refer to *Low Power Island Voice Activation – SVA 8.0* (80-40939-153) for resolution instructions.
C103	SVAUseCaseModifiedError	▪ Critical error: SVA QC use case is modified incorrectly. Solution: Refer to *Low Power Island Voice Activation – SVA 8.0* (80-40939-153) for resolution instructions.
E100	InvalidDataConnectionError	▪ Data connection is invalid ▪ Auto-fix: Remove the data connection
E101	InvalidControlLinkError	▪ Control connection is invalid ▪ Auto-fix: Remove the control connection
E102	InvalidSGPairError	▪ Subgraph pair is invalid ▪ Auto-fix: Remove the subgraph pair
E103	OffloadingNonSequenceModulesError	▪ User trying to offload non-sequence modules ▪ No auto-fix

Error Code	Error string	Description
E104	ModuleNotFoundError	▪ Module definition not found in the specified DSP ▪ No auto-fix
E105	ModulePortMissingError	▪ Module definition does not have input or output ports ▪ No auto-fix
E106	InvalidOffloadingOperationError	▪ User trying to offload modules from voice subgraph ▪ No auto-fix
E107	ContainerPropertyDataNotFoundError	▪ Container property missing ▪ Auto-fix: Add it with the default value
E108	InvalidContainerPropertyDataError	▪ Container property has out of range data ▪ Auto-fix: Correct it with the default value
E109	InvalidModuleCalibrationData	▪ Module calibration data is out of range ▪ Auto-fix: Correct it with the module default data
E110	InvalidModuleTagData	▪ Module calibration data for tag keys is out of range ▪ Auto-fix: Correct it with the module default data
E111	InvalidMultiChannelModuleDataError	▪ Multichannel module has different list of channels in different params ▪ No auto-fix
E112	InvalidModulePortpropertyError	▪ Module connection has a port number which is more than supported by the module definition ▪ No Auto-fix
E113	SPFPropertyDataNotFoundError	▪ SPF property is missing ▪ Auto-fix: Add with the default value
E114	InvalidSPFPropertyDataError	▪ SPF property data is out of range ▪ Auto-fix: Correct it with the default value
E115	MissingGSLPropertyError	▪ GSL property is missing ▪ Auto-fix: Add with the default value
E116	ToolPolicyError	▪ Cal data is present even if the tool policy is not supported ▪ Auto-fix: Remove the calibration data for that param
E117	ContainerNotFoundError	▪ Container is not found in the ACDB files ▪ No auto-fix
E118	ParamNotFoundError	▪ Param definition not found in the module definition ▪ Auto-fix: Remove the calibration data for that param
E119	Duplicate data connection	▪ Same data connection is added twice ▪ Auto-fix: Delete one data connection
E120	DuplicateControlLinkError	▪ Same control link is added twice ▪ Auto-fix: Delete one control link

Error Code	Error string	Description
E121	InvalidDanglingConnectionError	▪ Subgraph connection is added as dangling connection. Both subgraphs are in the same use case. ▪ Auto-fix: Remove the dangling connection and add a subgraph connection
E122	InvalidGSLPropertyDataError	▪ GSL property has out of range data ▪ Auto-fix: Correct it with the default data
E123	OutofRangePortIdError	▪ Module port property has a value greater than the max ports supported in the module definition ▪ No auto-fix
E124	SyncModuleCalidationError	▪ Sync Module's output connection(s) are missing for corresponding input connection(s)
E125	ToolPolicyValidationError_MissingData	▪ Calibration data is missing for its PID tool policy
E126	MailBoxIntentError	▪ Mailbox ctrl link has invalid intent
E130	InvalidGKVError	▪ A graph's GKV is invalid (conflict between switch ports). Such a graph will be assigned with zero GKV Solution: fix switch port KVs to make sure a routed graph has no conflict KVs (conflict means same key with different values).
E131	DuplicateGKVError	▪ Multiple graphs have same GKV. These graphs are assigned with zero GKV. Solution: go to zero GKV. Fix connections or/and switch port KVs to make sure each routed graph have unique GKV.
E132	InvalidInternalConnectionError	▪ A switch internal connection contains invalid port id Solution: this error means switch data is corrupted. Should delete the switch and add it again.
E133	SwitchPortKvsConflictError	▪ Switch has conflict KVs between its input and output ports (both KVs contains same key, but different values) Solution: update port KVs
E134	SwitchConcurrencyError	▪ A concurrent switch has invalid internal connection (refer to Concurrency) Solution: fix internal connections
E135	SwitchCtrlLinkError	▪ A switch has external ctrl link but no internal ctrl link for a ctrl port ▪ A switch has internal ctrl link but no external ctrl link for a ctrl port ▪ A switch's ctrl link intent does not match connected module Solution: Add correct ctrl link and fix intent

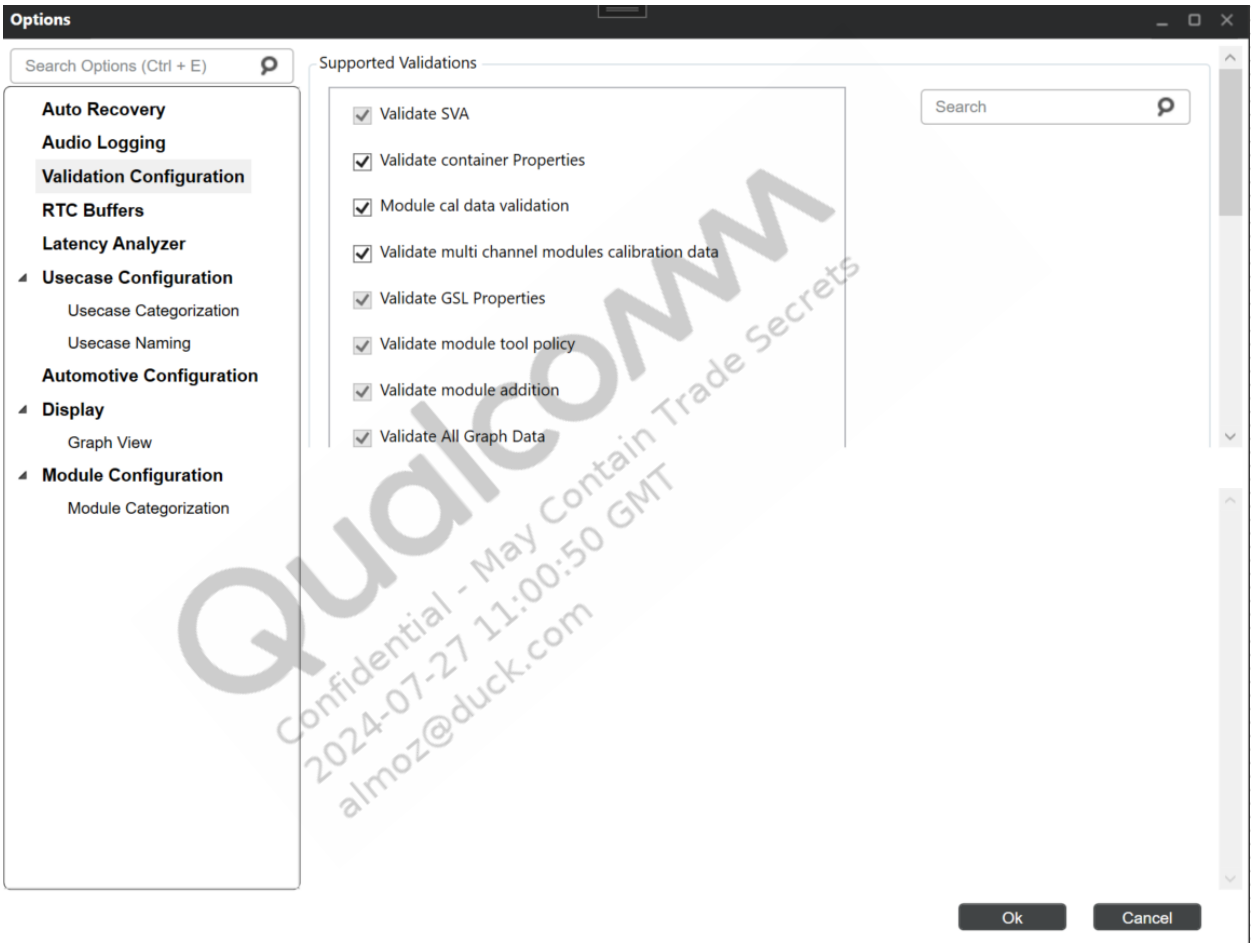
Error Code	Error string	Description
E141	MetaGraphRoutingError	▪ Route a graph's switches to create a GKV different than its current GKV. ▪ Route a graph's switches to find its GKV is conflict. Solution: this error normally means data is corrupted. Better to delete related subgraphs firstly. And redesign graphs from scratch.
E142	MetaGraphMatchingError	▪ A graph in ACDB file has corrupted data. Solution: this error normally means data is corrupted. Better to delete related subgraphs firstly. And redesign graphs from scratch.
E146	IndependentSwitchError	▪ An independent switch is not being used in any graph. Solution: AutoFix to delete it.
E147	MissingExternalConnError	▪ A switch does not have any input connection or output connection Solution: means this switch will not be in a routing path properly. Should add input or output connection correctly.
E148	SwitchPortMultiConnError	▪ A switch port has more than one connections. Solution: a switch port only can have one external connections. Delete all connections and only leave one.
E149	SwitchInternalConnError	▪ A switch with internal module, but it has a connection from input port directly to output port Solution: if a switch has internal module(s), all its internal connections should route to internal module(s).
E151	SwappedIidControlLinkError	▪ Between two subgraphs, a ctrl link exists twice but with source and destination iids swapped. Solution: AutoFix to delete one of them.
E152	DuplicateSGPairError	▪ Between two subgraphs, there are duplicated subgraph pair data with connection and/or ctrl link. Solution: AutoFix to delete one of them.
W100	SwitchRedundantConnectionError	▪ Switch has more than one input/output connections routing from/to same module and port, which is invalid. Solution: AutoFix to delete redundant connection(s).
W101	InvalidDataConnectionWarning	▪ A module's data port has more than one connections. Solution: This is a warning. Sometimes, this is expected. If not, should remove connection and add connection with different port of the module.

Error Code	Error string	Description
W102	NoEndPointGraphWarning	▪ A graph has no source EP subgraph or sink EP subgraph. Solution: This is a warning. Sometimes, it is expected. If not, should correct the subgraph(s) by adding EP module.
W103	ZeroGKVWarning	▪ There is a zero gkv graph Solution: Zero GKV graph is not a valid graph. It will be useless on target. Should either delete it or update it to correct GKV.
W104	NotConnectedGraphWarning	▪ A graph contains at least one subgraph without connection with any other subgraph. Solution: Normally this is not valid. Should connect subgraph correctly.
	InvalidContainerStacksizeDataWarning	▪ A container's Stack Size property is not correct by checking with its modules. Solution: AutoFix to fix it.

4.12.3 Select validations

To select which validations to run:

1. Click **Edit->Options.**
2. Select the Validation Configuration tab.
3. Uncheck the optional validations which are not required.



Users can select any validation to see what it does and what types of errors it will throw. Users can unselect only optional validations here if they do not want to run those validations during save/save as. Mandatory validations cannot be unchecked and QACT always runs these validations.

4.12.4 Run all validations

1. Click **Edit->Options.**
2. Select the Validation Configuration tab.
3. Close the Options window.
4. Save the ACDB files.

4.12.5 Validation errors window

After running validations, any validation errors that are found are shown in the validation errors window at the bottom of the QACT screen. Users can do the following in validation errors window:

- Click **Go to Error** to navigate to the error in the graph view
- Click **AutoFix** to fix the issue in the ACDB files. The AutoFix button is only present for errors QACT can fix. Some errors do not have this capability.
- Right-click any error and ignore that single error or all the errors with that error code. Once the users ignore the errors, Validation Manager will ignore those errors and will not show them again.
- Export the details of the errors to text format
- Filter errors by error code
- Search for a specific error using the search box
- Right-click and clear all errors in the validation errors window
- Right-click any error and delete that specific error from the validation errors window
- Right-click any error and copy it
- Multi-select using the Ctrl key and right-click to copy, auto-fix, ignore, or delete the selected items

4.13 H2XML Annotations

To prepare H2XML definitions, H2XML annotations should be added in header files and converted to H2XML format xml files that can be imported into ACDB through QACT's Discovery Wizard. This section describes how H2XML annotations should be used and how they work in QACT. There are several types of H2XML definitions. Each of them have different sets of annotations and are used in different features of QACT. All header files with H2XML annotations should be converted to xml files. H2XML.exe (command line tool) should be used to do conversion with following format:

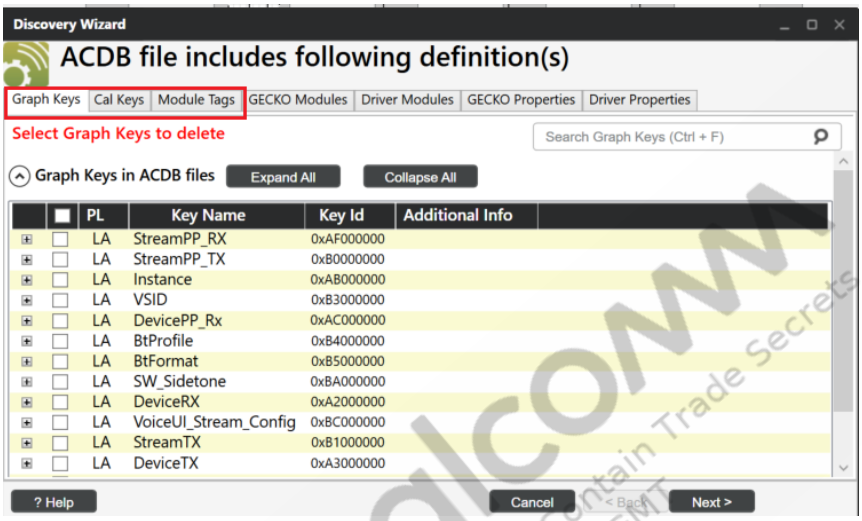
```
h2xml.exe –conf <config_file> -o <output_file> [optional parameters] -t
<config type> input_file
```

"-t <config type>" decides what type of definition input_file is. There are five types of definitions supported:

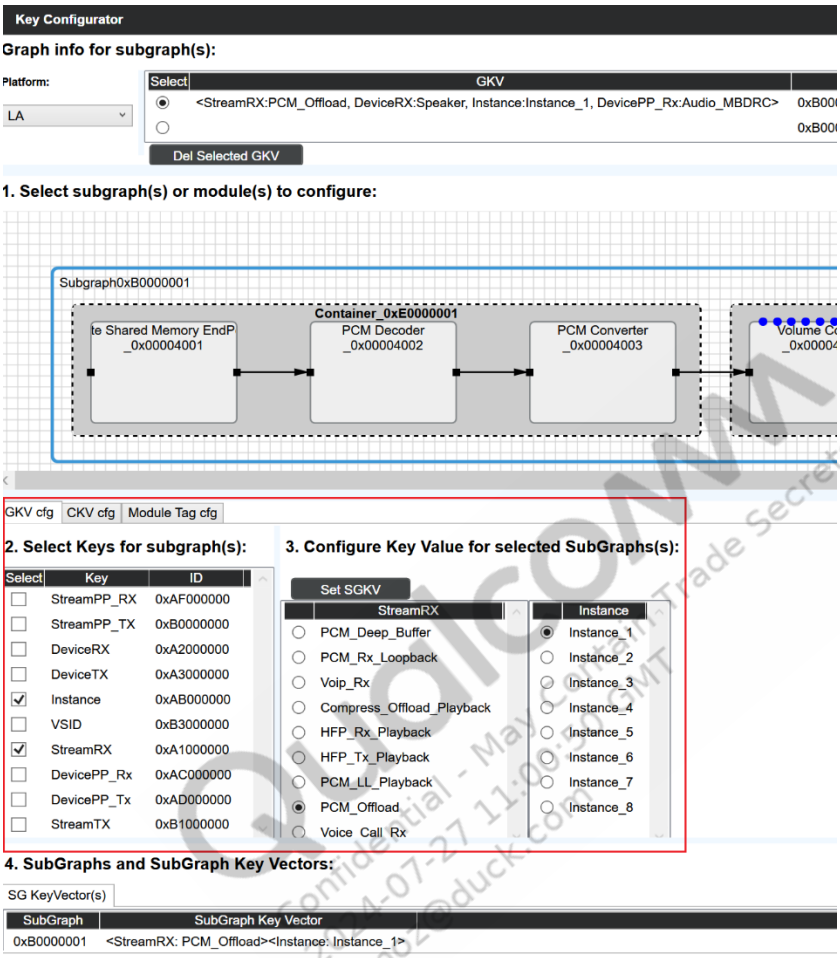
- Key
- spfModule
- DriverModule
- spfProp
- driverProp

4.13.1 Key and module tag definitions

Key and module tag definitions are defined by platforms. Different platforms may have different sets of definitions. These definition's file's config type is Key. Key definitions are common definitions which can be used as Graph Keys, Calibration Keys and Module Tags with different annotations. After importing Key and module tag definitions into ACDB, they can be seen in Discovery Wizard -> View Defs in ACDB.



These definitions are used in Key Configurator when the system designer creates a graph by adding GKV, CKV, and module tags/TKVs.



The following is an example with all annotations for key and module tag definitions in a .h file"

```
enum AllKeyIds{
    Samplerate = 0xA0000000, /**< @h2xmle_name{srate} */
    Bitwidth = 0xB0000000,
    Channel = 0xC0000000,
};

/**
    @h2xmlk_key {Samplerate }
    @h2xmlk_sampleRate
    @h2xmlk_description {Description 10}
*/
enum Key_Samplerate {
    samplerate1 = 0xA0000001, /**< @h2xmle_sampleRate{8000}
@h2xmle_description {Description 101}*/
```

```
    samplerate2 = 0xA0000002, /**< @h2xmle_sampleRate{16000}
@h2xmle_description {Description 102}*/
};

/**
    @h2xmlk_key {Bitwidth}
    @h2xmlk_description {Description 20}
*/
enum Key_Bitwidth {
    bitwidth1 = 0xB0000001, /**< @h2xmle_description {Description 201}*/
    bitwidth2 = 0xB0000002, /**< @h2xmle_description {Description 202}*/
};

/**
    @h2xmlk_key {Channel}
    @h2xmlk_description {Description 30}
*/
enum Key_Channel {
    channel1 = 0xC0000001, /**< @h2xmle_description {Description 301}*/
    channel2 = 0xC0000002, /**< @h2xmle_description {Description 302}*/
};

/**
    @h2xmlk_gkeys
    @h2xmlk_description {Description 40}
*/
enum Graph_Keys {
    gk_samplerate = Samplerate,
    gk_bitwidth = Bitwidth
};

/**
    @h2xmlk_ckeys
    @h2xmlk_description {Description 50}
*/
enum Cal_Keys {
    ck_samplerate = Samplerate,
    ck_bitwidth = Bitwidth,
};

#define ModuleTag1 0xE0000000
#define ModuleTag2 0xF0000000
/**
    @h2xmlk_modTag {"ModuleTag1", ModuleTag1}
    @h2xmlk_description {Description 60}
    @h2xmlk_isVoice
*/
enum ModuleTag1_Keys {
```

```
    tk1_samplerate = Samplerate,
    tk1_bitwidth = Bitwidth
};

/**
    @h2xmlk_modTag {"ModuleTag2", ModuleTag2}
    @h2xmlk_description {Description 70}
*/
enum ModuleTag2_Keys {
    tk2_samplerate = Samplerate,
    tk2_channel = Channel
};
```

In this example:

- @h2xmlk_key – This annotation defines an enum as a key. Each enum element will be a value of the key.
- @h2xmlk_gkeys – Indicates all elements of an enum are graph key IDs. These key definitions must also be defined with @h2xmlk_key.
- @h2xmlk_ckeys – Indicates all elements of an enum are cal key IDs. These key definitions must also be defined with @h2xmlk_key.
- @h2xmlk_modTag – Indicates all elements of an enum are keys of a module tag. These key definitions must also be defined with @h2xmlk_key.
- @h2xmlk_sampleRate – This is a special annotation for a cal key which is intended to be used as a sampling rate. It is special because sampling rate is used by some tuning tools (MBDRC, IIR, etc.). If such a cal key is annotated, QACT will be able to link its values to tuning tools.
- @h2xmle_sampleRate – Indicates the sample rate for the SampleRate key's value. If a key is annotated with @h2xmlk_sampleRate, all its key values should have this annotation so that its values can be mapped to the correct sample rates understandable to QACT.
- @h2xmlk_isVoice – Indicates a key or module tag is only for voice use cases. A voice key or module tag can only be assigned to voice subgraphs and modules. In Key Configurator, for non-voice subgraph and modules, voice keys and module tags will not be shown.

4.13.2 Module definitions

A module definition mainly contains two parts – The module's meta information and its PID information. There are two types of modules:

- SPF modules – These modules are the modules in a use case (for example, volume module, IIR module etc.). Their data can be tuned from Graph Designer directly. The config type is spfModule.
- Driver Modules – These modules are used on platform software and are independent to use cases (for example, the CodecProfile module). They can be defined by the platform to satisfy the platform's software requirements. Their data can be accessed and tuned from Tools- >Driver Module. Such a module does not need module meta information in its definition. The config type is DriverModule.

The following is an example with all annotations for a module definition in a .h file:

```
 /** Module ID for Gain module  */
#define MODULE_ID_GAIN              0x07001002
/** @h2xmlm_module       {"MODULE_ID_GAIN",
                          MODULE_ID_GAIN}
    @h2xmlm_displayName  {"Gain Module"}
    @h2xmlm_description  {Gain Module }
    @h2xmlm_dataMaxInputPorts        { 1 }
    @h2xmlm_dataInputPorts           { IN  = 02}
    @h2xmlm_dataOutputPorts          { OUT = 01}
@h2xmlm_dataMaxOutputPorts      { 1 }
@h2xmlm_ctrlStaticPort          {cntrlPort0 = 0,
                                 intent1=1,
                                 intent2=2}
            @h2xmlm_ctrlStaticPort          {cntrlPort1 = 1,
                                 intent3=3}

        @h2xmlm_ctrlDynamicPortIntent   { intent1=2,maxPorts=5 }
    @h2xmlm_ctrlDynamicPortIntent   { intent2=7,maxPorts=3 }
    @h2xmlm_supportedContTypes      {APM_CONTAINER_CAP_PP, APM_CONTAINER_CAP_CD,
APM_CONTAINER_CAP_EP}
    @h2xmlm_stackSize               { 4096 }
    @{                  <-- Start of the Module -->
*/

/* ID of the parameter used to set the gain */
#define PARAM_ID_GAIN           0x08001006

/** @h2xmlp_parameter   {"PARAM_ID_GAIN",
                         PARAM_ID_GAIN}
    @h2xmlp_description {Configures the gain}
    @h2xmlp_toolPolicy  {Calibration; RTC} */

#include "gk_begin_pack.h"
```

```
/* Payload for parameter param_id_gain_cfg_t */
struct param_id_gain_cfg_t
{
      uint16_t gain;
 /**< @h2xmle_description {Linear gain (in Q13 format)}
      @h2xmle_dataFormat  {Q13}
      @h2xmle_default {0x2000} */

       uint16_t reserved;
  /**< @h2xmle_description {Clients must set this field to 0.}
       @h2xmle_rangeList {"0"=0}
       @h2xmle_visibility     {hide} */
}
#include "gk_end_pack.h"
;
/* Structure type def for above payload. */
typedef struct param_id_gain_cfg_t param_id_gain_cfg_t;
/**  @}                   <-- End of the Module -->*/
```

4.13.2.1 Module meta info annotations

Module meta info annotations are used to define a module's meta info, which can be used in a graph. The following module meta info annotations are available:

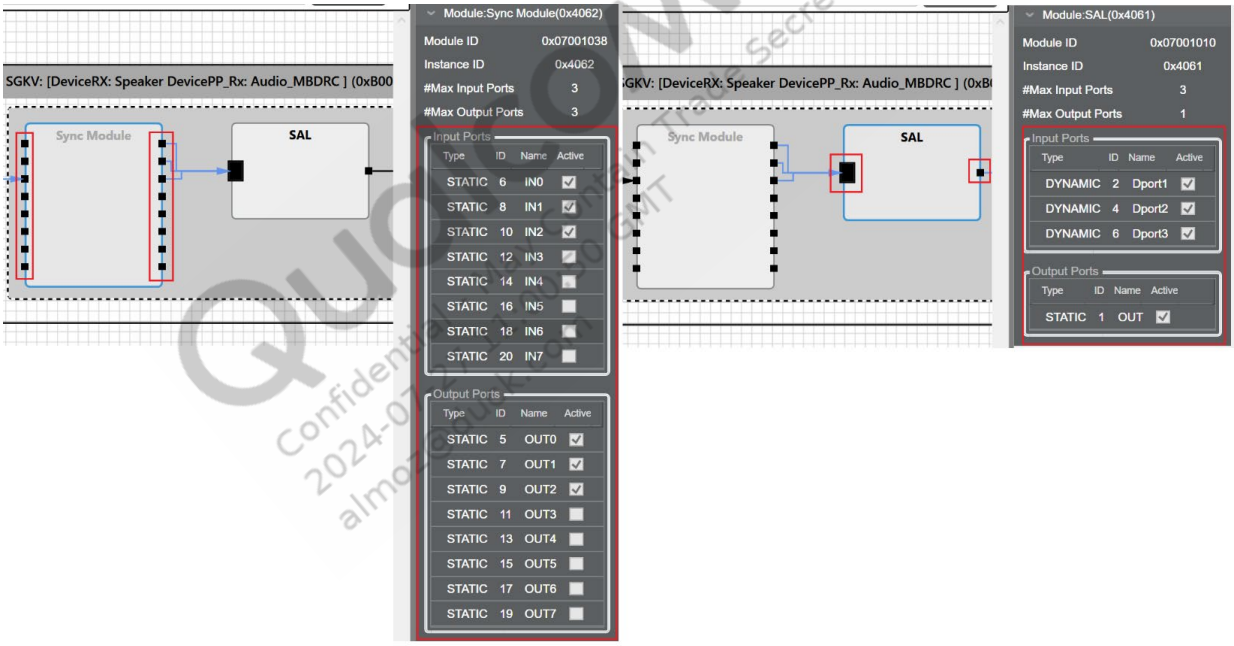
- @h2xmlm_module {<module name>, <module ID>} – This annotation defines a .h file and contains a module definition. It contains a module's name string and module ID. After this annotation, all annotated parameters encapsulated between @{ and @} are considered PIDs for this module.
- @h2xmlm_dataMaxInputPorts {<max input port number>} – This annotation defines a module's max number of input data ports.
- @h2xmlm_dataInputPorts {<port1 label>=<port1 ID>; <port2 label>=<port2 ID>; } – This annotation defines a module's static input port(s). It should contain a port label (name) and port ID. If more than one static ports are supported, they can be added with ";". The number of static ports must be the same or less than the max input port number.

  If this annotation exists, the defined port(s) are shown statically on Graph Designer. If there is no static port defined, this indicates that all the ports (less than the max port number) are dynamic ports.
- @h2xmlm_dataMaxOutputPorts {<max output port number>} – This annotation defines a module's max number of output data ports.

- @h2xmlm_dataOutputPorts {<port1 label>=<port1 ID>; <port2 label>=<port2 ID>; } – This annotation defines a module's static output port(s). It should contain a port label (name) and port ID. If more than one static ports are supported, they can be added with ";". The number of static ports must be the same or less than the max output port number.

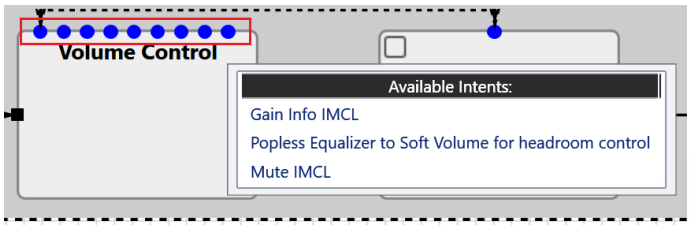
  If this annotation exists, the defined port(s) are shown statically on Graph Designer. If there is no static port defined, this indicates all the ports (less than the max port number) are dynamic ports.

  Data ports are shown on left and right sides of a module in Graph Designer. Their information is also shown in property view when a module is selected. In the following example, the module "Sync Module" has 8 static input ports and 8 static output ports defined. Its max input and output ports are 8. The other module, SAL, has no static input port defined. All its input ports are defined as dynamic since the @h2xmlm_dataInputPorts annotation is not added. It has one static output port added.



- @h2xmlm_ctrlStaticPort {<port1 label>=<port1 ID>, <intent1 label>=<intent1 ID>, <intent2 label>=<intent2 ID>, …} – This annotation defines one static control port for a module. It contains the port label/name, port ID, intent label/name and intent ID. More than one intents can be added. If more than one static control port is needed for a module, this annotation can be added multiple times.
- @h2xmlm_ctrlDynamicPortIntent {<intent label>=<intent ID>, maxPorts=<max port number>} – This annotation defines the dynamic port(s) of a module for one intent. It should contain one intent label/name and intent ID. It should also define maxPorts. One annotation is used for one intent. If a module supports multiple intents with dynamic control ports, this annotation can be added multiple times.

Control ports are shown on top of a module in Graph Designer. Each control port's available intents are shown when the cursor is put on the port (one control link will make one available intent occupied) . In following example, the Volume Control module contains 9 control ports. When the cursor is on the last port, a floating window shows the port has three intents available. When connecting two modules via their control ports, only ports with the same available intent can be connected.



- @h2xmlm_supportedContTypes {<Container type 1 ID>, <Container type 2 ID>, … } – This annotation defines the supported container type(s) for a module. Container types are defined as one of the container properties. A module can only be added into a container with the same container type it supports.
- @h2xmlm_stackSize {<stack size of a module>} – This annotation defines a module's stack size.

4.13.2.2 PID annotations

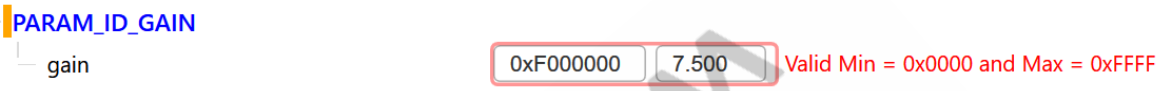
The following PID annotations are available:

- @h2xmlp_parameter {<parameter name>, <parameter id> } – Defines a parameter's name and ID. The C structure following this annotation is a data structure of this parameter ID. More than one policy can be used for one parameter.
- @h2xmlp_toolPolicy {<policy1>, <polcy2>, … } – Defines a parameter's policy in ACDB. QACT uses these policies to control how a PID's payload should work in different modes. The following are the supported policies:
  - Calibration – PID's payload will be stored in ACDB. The PID can be shown in QACT under non-connected mode.
  - RTC – PID's payload will be available only in RTC mode. The PID can be shown in QACT under RTC mode.
  - RTM – PID's payload will be queried and shown only in RTM mode.
  - RTC_READONLY – PID's payload is read-only in RTC mode.
  - NO_SUPPORT – PID is not supported. Will not exist in any mode.

  If this annotation is not added for a parameter, QACT treats the parameter's policy as Calibration and RTC by default.

Inside a parameter C structure, each element can have its own annotations. Refer to *H2XML User Guide* (80-VM407-19) for details of all supported annotations. The following are some useful annotations for elements:

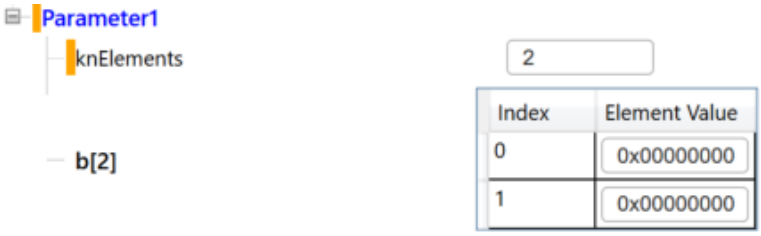
- @h2xmle_default {<default value>} – Indicates the default value of an element. The default value can be decimal or hex. When a module is added to a use case, all its PID's payloads are created based on each element's default value.
- @h2xmle_range {<min>..<max>} – Indicates the range of an element's value. Min and max can be decimal or hex. When tuning an element's value, QACT uses this range to do a safe check. If an input value is out or range, QACT shows a warning as shown in the following example.



- @h2xmle_rangeList {<val1 name>=<val1>; <val2 name>=<val2>; … } – Indicates the supported list of values for an element. For such an element, QACT will show its values in a dropdown box in Calibration View. In the following example, the element "enable" is annotated as @h2xmle_rangeList {"Enable"=0x1; "Disable"=0x0}.

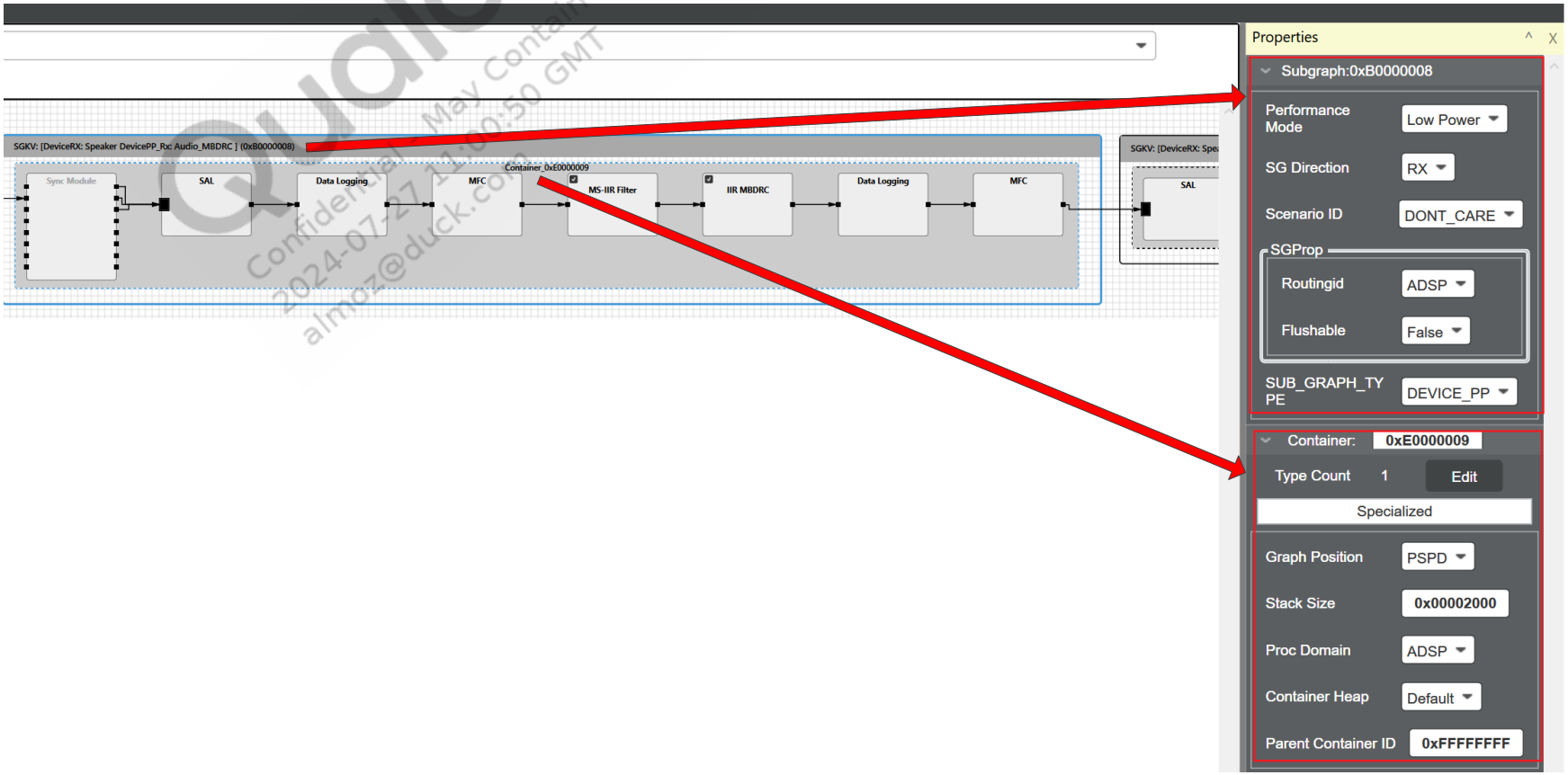
- @h2xmle_variableArraySize {<element name> } – Defines a dynamic array. The array's length has dependency on the value of another element in same parameter. In the following example, element b is defined as a dynamic array. Its array length can be decided by the value of knElements. In QACT Calibration View, if the knElements value is changed, b array length will be changed automatically. In this example, b is defined as an integer array. It is also supported if it is defined as a struct array.

```
/** @h2xmlp_parameter   {Parameter1,0x11111199}*/
struct param_1{
          unsigned short knElements;
          int b[0];
     /**<
          @h2xmle_variableArraySize {knElements}
     */
};
```



4.13.3 Property definitions

There are two types of property definitions – spfProp and driverProp. These definitions are defined by the SPF software and platform software and have dependency on the software implementation in a build. QACT users are not expected to define/modify these definitions. Their information is shown in QACT -> Property Window as follows.



A subgraph contains both spfProp data and driverProp data, while a container contains only spfProp data. All are shown if a subgraph or container is selected.

4.14 Distributed ACDB support

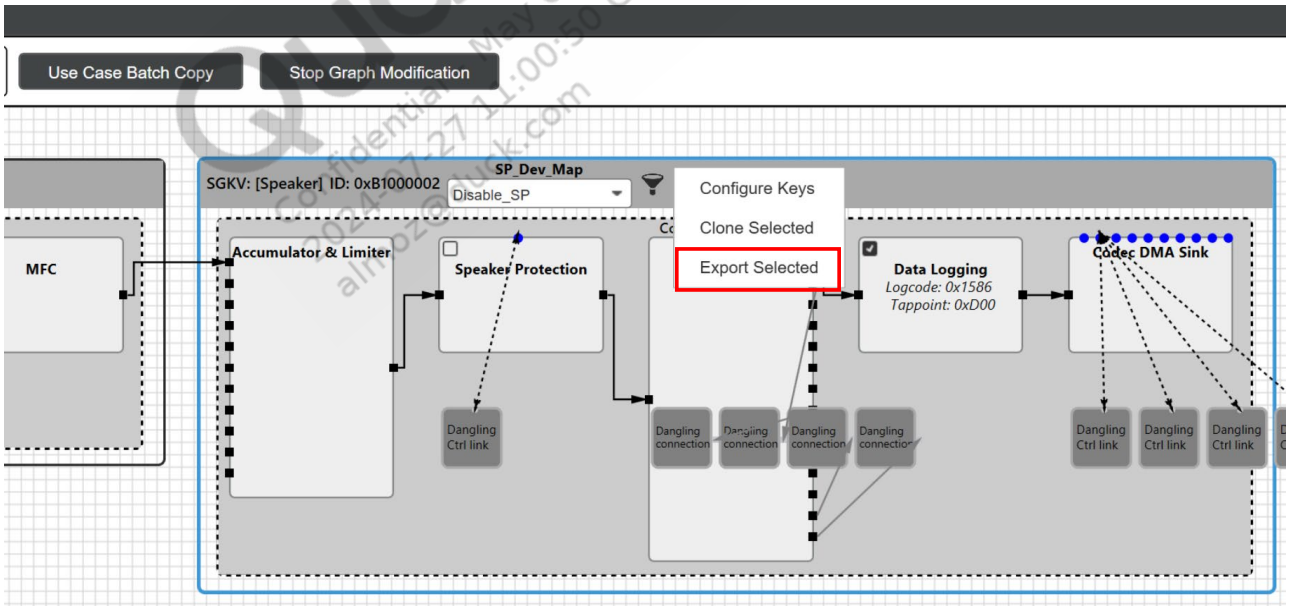
For Automotive platforms, some users have more than one virtual machine (VM). Each VM has its own ACDB with a unique VMID. In ACDB, VMID is set as the second MSB digit of all IDs in hexadecimal format for subgraphs, containers, and module instances. For example, if a VM's VMID is **1**, in its ACDB, its subgraph ID should be 0xB**1**xxxxxx, its container ID should be 0xE**1**xxxxxx, and its module instance ID should be 0x0**1**xxxxxx.

VMs can share subgraph(s) by exporting and importing from/to their ACDB

- A VM can export a subgraph which can be imported to other VM's ACDB
- A VM can import a subgraph which is exported from another VM's ACDB
- Since the VMID is unique, an imported subgraph and its container(s) and module instance(s) will have different second MSB digit than other subgraphs, containers and module instances in a VM ACDB.

4.14.1 Export subgraph

In the Graph Designer UI, after clicking **Start Graph Modification**, select a subgraph, right-click, and select **Export Selected**. Click it to export the selected subgraph to a *.ssg" file.

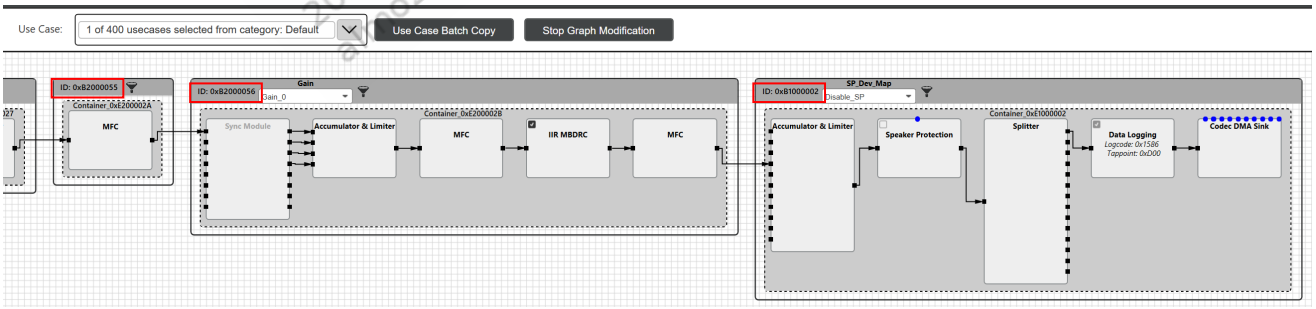


4.14.2 Import subgraph

A VM ACDB can import a subgraph exported from another VM. **Go to Tools -> Import Shared Subgraph Wizard** and import a *.ssg file. A summary will be shown. All imported subgraphs will be added to the Zero GKV graph by default. To use it, select the Zero GKV from UseCase selection dropdown box.

In Graph Designer, an imported subgraph's contents (subgraph, container, and module properties, connections, calibration data) are always read only after **Start Graph Modification** is clicked. It is only allowed to be connected with other subgraphs.

The following example shows a subgraph 0xB100000002 (its owner VMID is 1) that has been imported to a VM's ACDB with VMID 2. It is connected to subgraph 0xB20000056 to create a new graph.

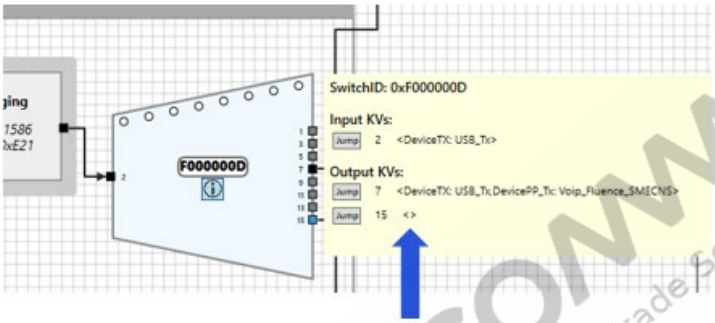


The only data that can be modified for an imported subgraph is a dangling link. A new added dangling link for a subgraph will change the subgraph's data. When this happens, the imported subgraph should be exported again to include the new added dangling link. The owner of the subgraph knows this; the owner VM should import this updated subgraph back to keep its data consistent across VMs.

4.15 Switch Port KV info control

Switch port KV info control is visible for each switch to view the input and output port KVs. The control consists of switch ID, input and output KV strings along with port IDs.

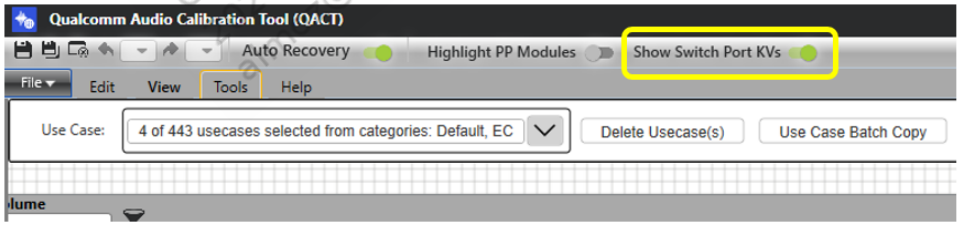
- Only the ports that have active connections in graph view are displayed.
- The Jump button is present to highlight the subgraph to/from which the connection at this port goes.
- If there is no KV for any port, it simply shows “<>” for that port.



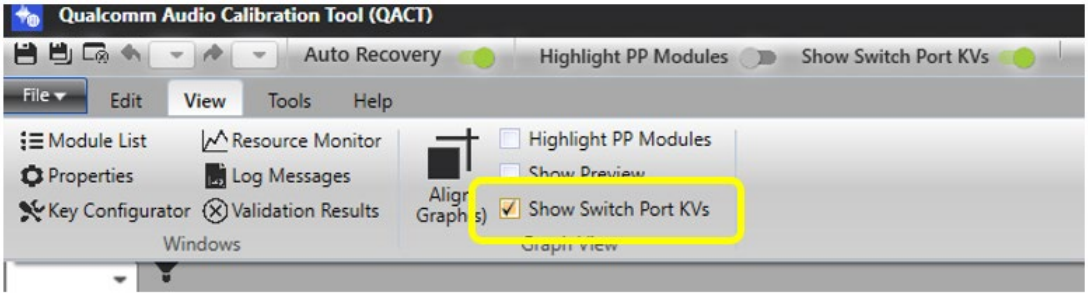
- There needs to be at least one connection to/from the switch to see the switch port KV info control.

4.15.1 Options to enable switch port KV info

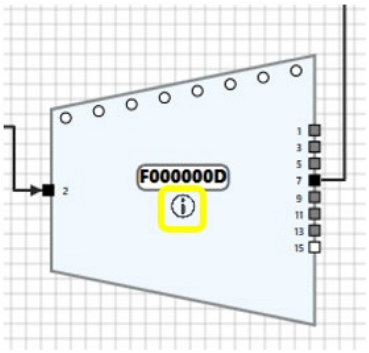
- Toggle button – Shows/hides the switch port info control for all the switches in the graph view



- Menu item – Shows/hides switch port info control for all the switches in the graph view

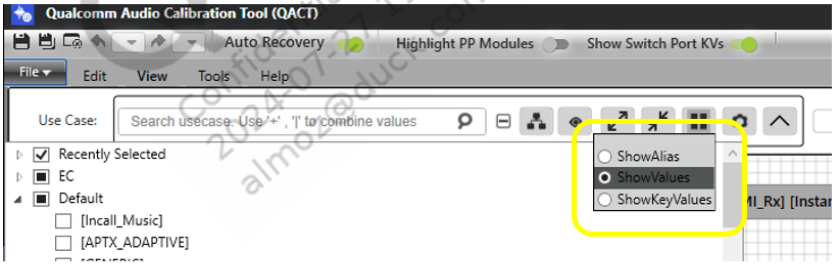


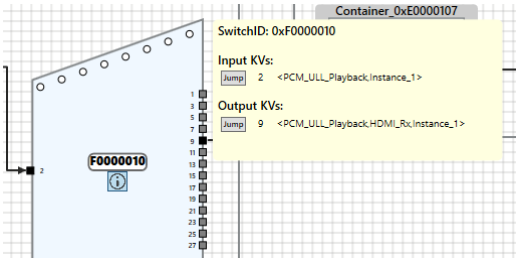
- Info button for each switch
  - Info icon on switch acts as toggle button
  - Shows/hides switch port info control for the switch
  - Switch port KV info control can be shown or hidden irrespective of toggle button using the switch info icon

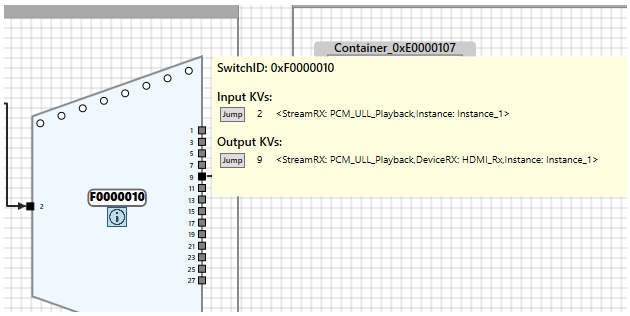


4.15.2 Display of KV information

- The KV string at a particular port
- There can be multiple KVs at a port
- User can set the display format of KV (that is, show only values or key-values) in the use case selection setting

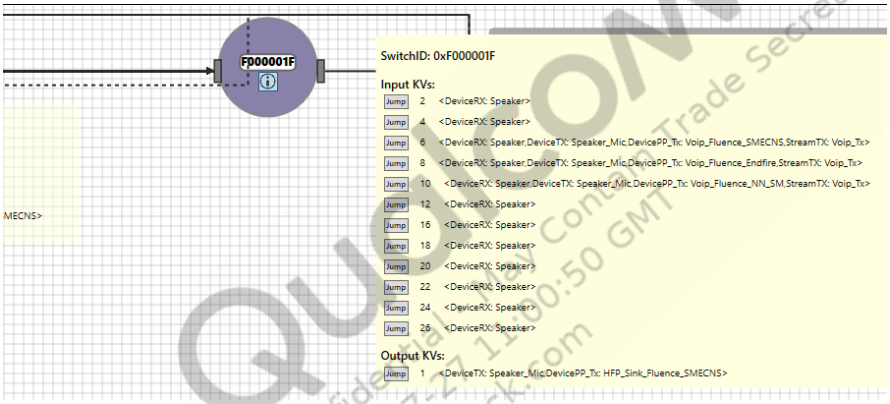






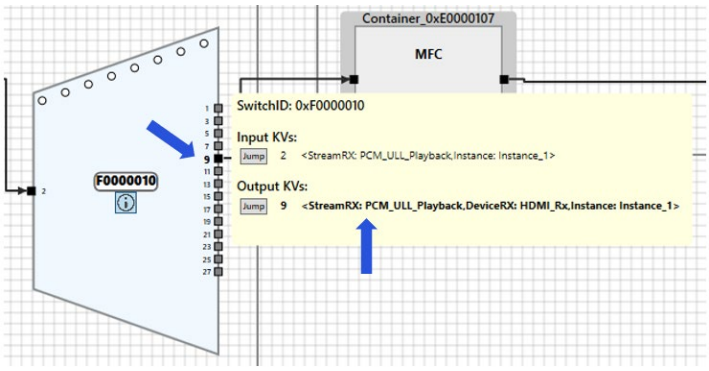
4.15.3 EC switches

For EC switches, the connections and ports are not visible like other switches. Hence, the KV information is obtained from internal connections and displayed in the switch port KV information control.



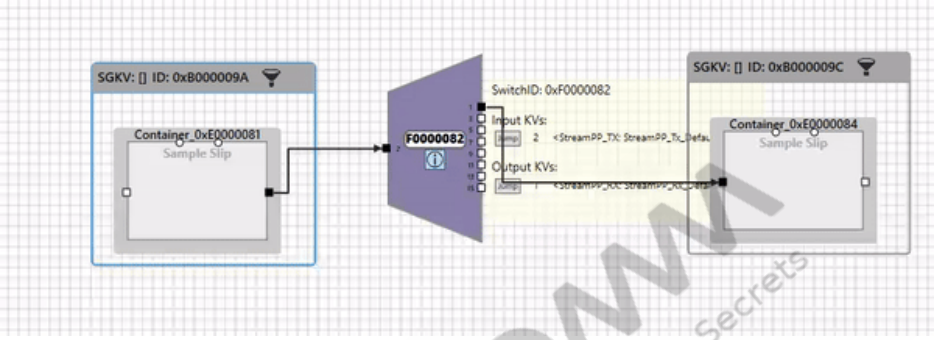
4.15.4 Highlighting any row in switch port KV info control

- When the mouse moves on top of any row, the port ID and KV are displayed in bold and the corresponding port label in switch is highlighted
- Also happens vice-versa (that is, hover on the port label and the corresponding port row is highlighted in Switch Port KV Info control)



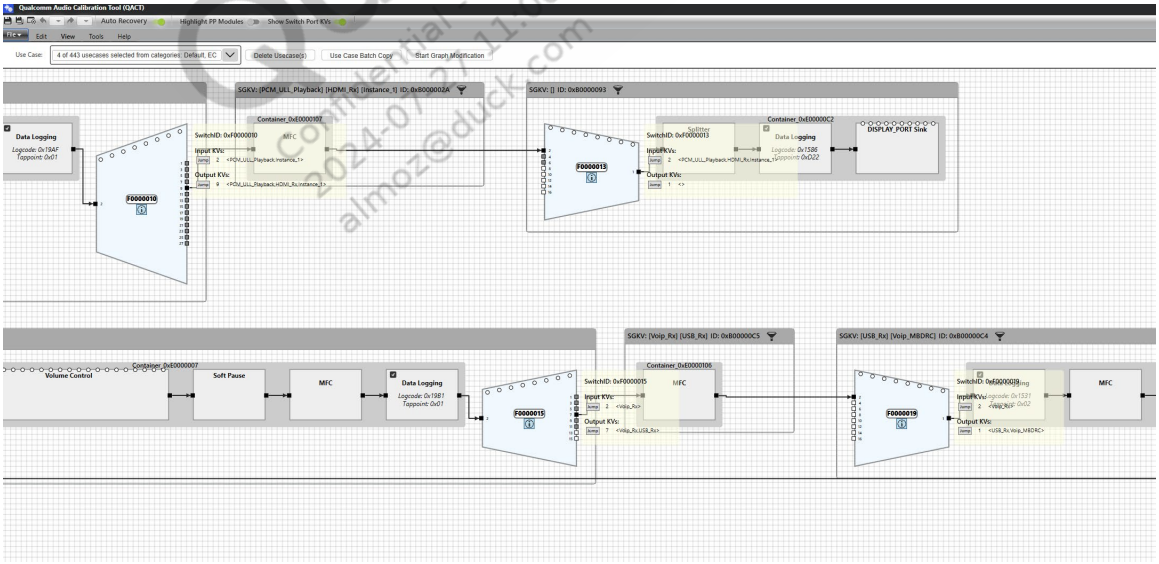
4.15.5 Jump to other end of subgraph or independent switch

- The jump button for each port row in the Switch Port KV Info control selects/highlights the subgraph to which the connection at this port connects to or from which the connection starts
- The other end can be an independent switch too, which will be highlighted when Jump is clicked.

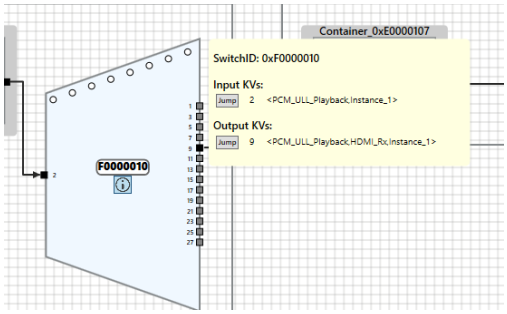


4.15.6 Visibility states of switch port KV info control

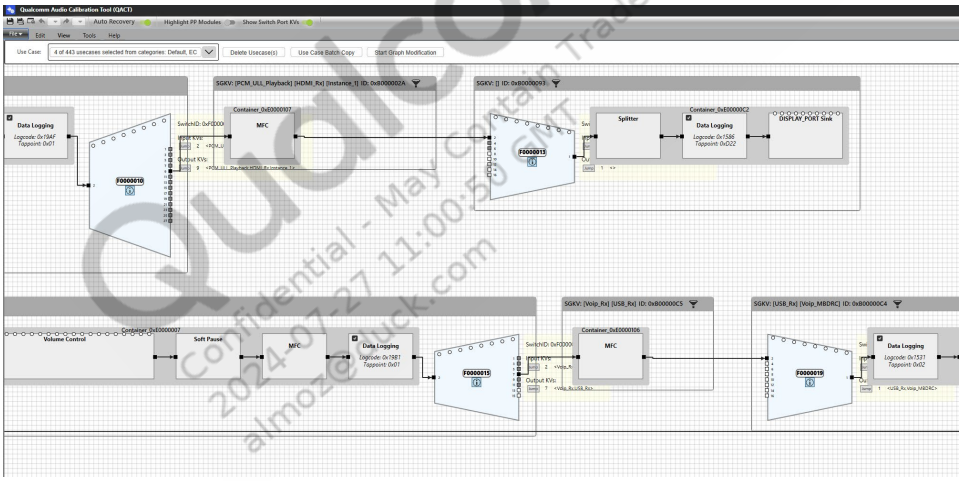
- Initial/half transparent state – When the visibility is set in the toggle button or menu item, the controls are visible with half transparency. This way, other controls like modules or connections under this control and visible partially.



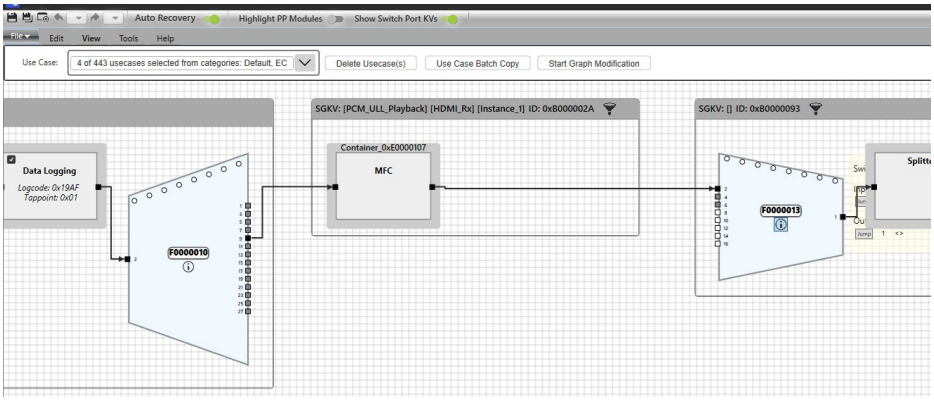
- Fully visible state – User needs to click on a particular control or switch info icon to make it fully focused and visible



- Partially visible state
  - When clicking outside of the control (that is, anywhere else in graph view), the control is placed at the bottom. This will allow users to access any modules/connections underneath.
  - To bring it to the fucus, click on the control or on the switch info icon.



- Collapsed state.
  - When visibility is disabled via toggle button or menu item, the controls are collapsed
  - When the switch info icon is clicked, it collapses the control



- Switch port KV info control can be shown or hidden irrespective of toggle button using the switch info icon

- Graph modification state
  - When start graph modification is clicked, the toggle button is disabled. All switch port KV info controls that were visible are now collapsed.
  - To view the control back, the user can click on the info icon on the switch
- Modification of switches
  - Adding a new switch – When a new switch is added, the switch port KV info control is not immediately visible. It needs at least one active connection in graph view to be visible.
  - Deleting a switch – When a switch is deleted the switch port KV info control is removed too
  - Adding/removing switch connection – Switch port KV info control is updated accordingly
  - Updating the KV of switch ports – The updated KV reflects in the switch port KV info control.

4.16 Integrated resource monitor (IRM)

QACT provides the capability to monitor resources on-device after connecting in online mode. The tool provides monitoring of several capabilities like processor metrics, container metrics, and module metrics depending on the support present on device. QACT also provides capability to record the metrics and save them for offline purposes.

NOTE: IRM does not work on production builds.

The following table provides a list of metrics supported by QACT.

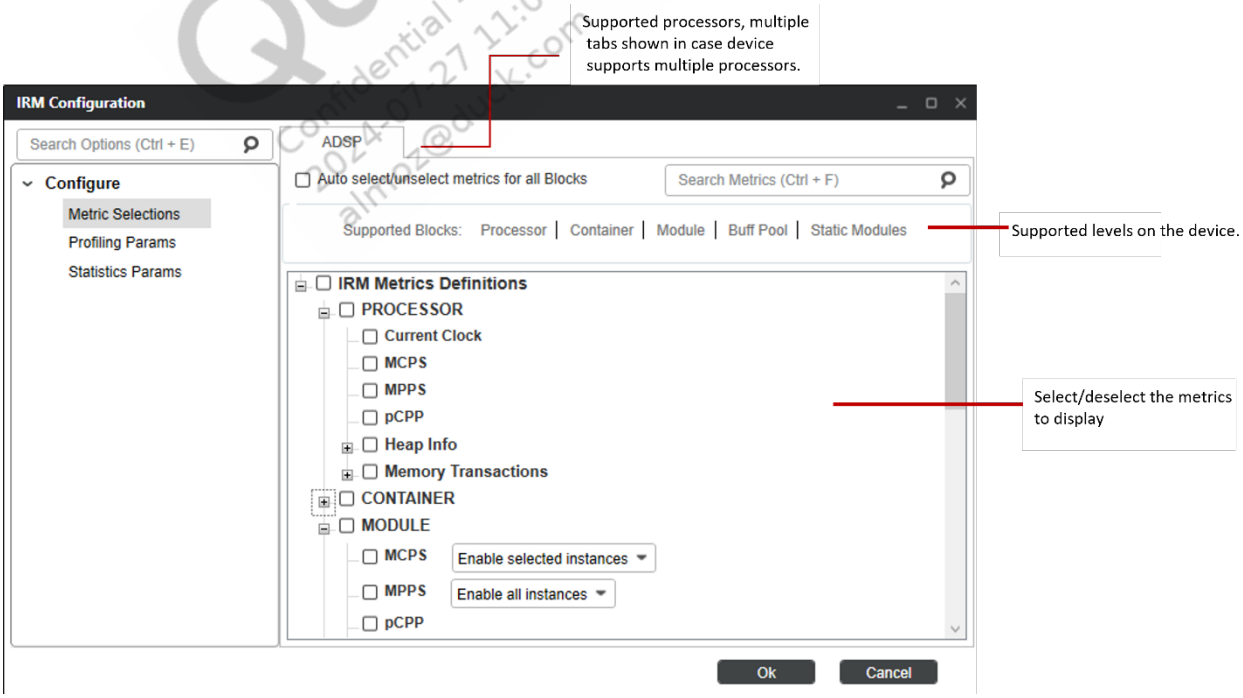
	NLPI - Chip Level	**LPI - Chip Level**	**NLPI – Container (Use Case part)**	**LPI – Container (Use Case part)**	**NLPI - Module**	**LPI – Module**	**Buffpool**	**Static modules**
Million cycles per second (MCPS)	Yes	Yes	Yes	Yes	Yes	Yes	NA	Yes
Million packets per second (MPPS)	Yes	Yes	Yes	Yes	Yes	Yes	NA	Yes
Processor cycles per packet (pCPP)	Yes	Yes	Yes	Yes	Yes	Yes	NA	Yes
Current clock	Yes	Yes	NA	NA	NA	NA	NA	NA

	NLPI - Chip Level	LPI - Chip Level	NLPI – Container (Use Case part)	LPI – Container (Use Case part)	NLPI - Module	LPI – Module	Buffpool	Static modules
Static memory (code + data)	Not possible	Not possible	Not possible	Not possible	Not possible	Not possible	Not possible	Not possible
Heap Info (RAM)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Read/Write BW	Yes	Yes	NA	NA	NA	NA	NA	NA

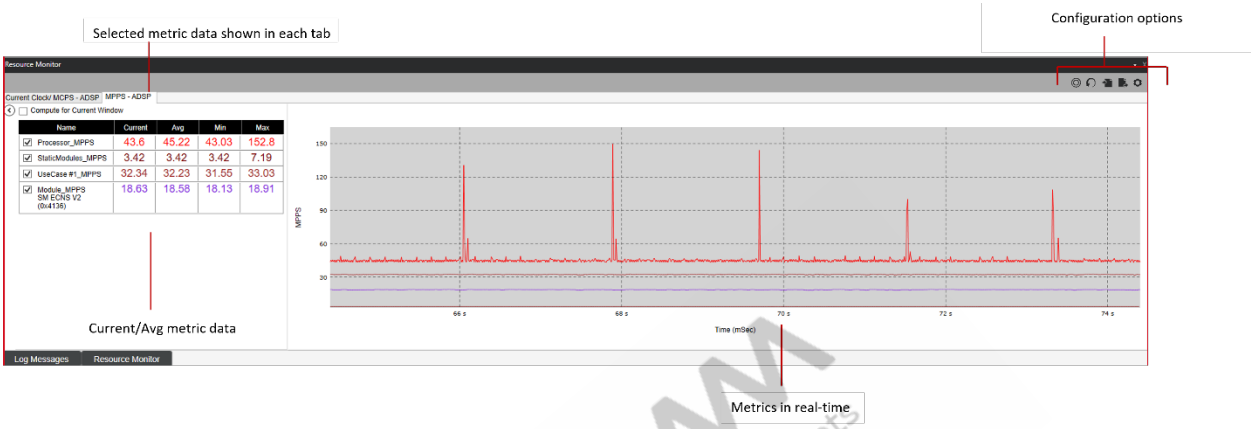
4.16.1 Monitoring resource metrics in online mode

To monitor metrics:

1. Connect QACT to target.
2. Click **Tools -> Resource Monitor Configuration.**
3. Select **Metrics Selection** on the left.
4. Select the metrics to monitor on the right.
5. Click **OK** to close the window and start monitoring.



Once metrics are selected, QACT shows the data as shown below. For monitoring metrics of specific module/container, the user needs to select the module in real time mode.



NOTE: QACT also supports grouping related data (for example, heap info, memory transactions, etc.)

4.16.2 Export/import configured metrics

Once the required metrics are configured and the required containers/modules are selected, click **Export Config** to export the selected list of metrics and containers/modules. It can later be imported in another session to automatically select the selections made during previous session. Users can the use this feature to share the config files and record required metrics for analysis.

4.16.3 Record and save metrics

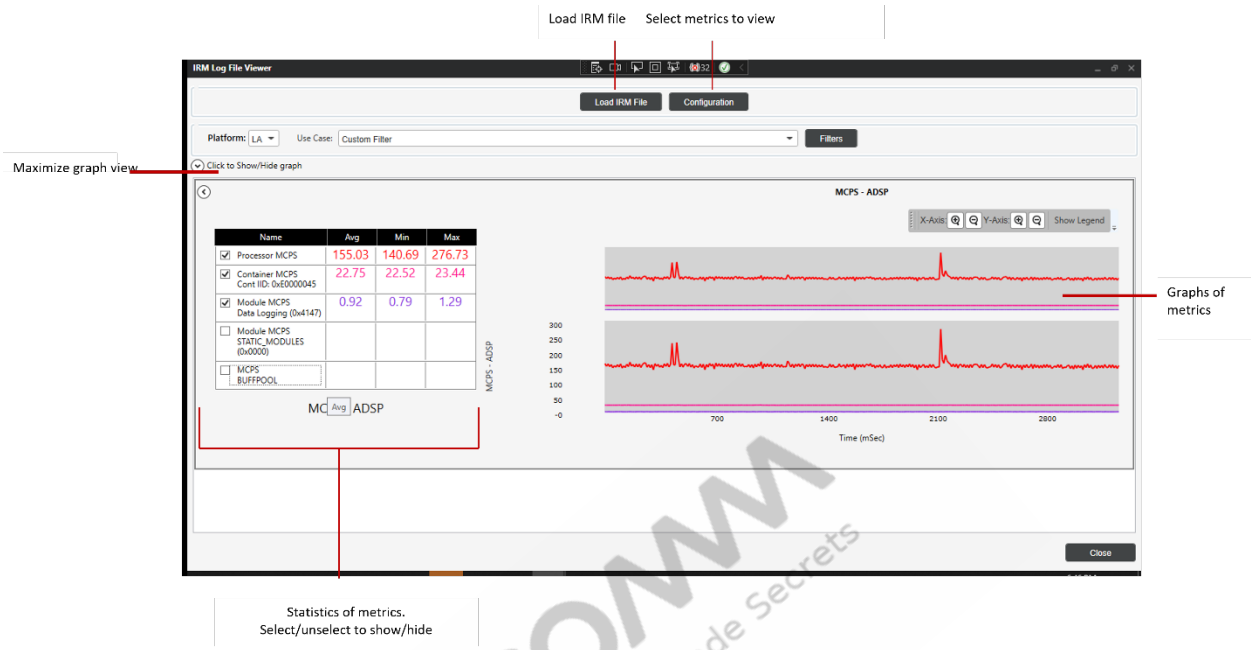
Once the required metrics are configured and the required containers/modules are selected, click **Start Recording** to start capturing data. Click the button again stop recording. Select the file path on pop-up to save the recorded file.

4.16.4 Offline mode

QACT supports the ability to load recorded IRM log file in offline mode. To view the IRM log file in offline mode:

1. Open an ACDB file in offline mode.
2. Click **Tools -> Load IRM Log**.
3. Select the recorded .irm file.
4. Click **Configuration** to select metrics. By default, this view will be open after parsing the IRM file.
5. Select/unselect metrics on left to show/hide the graphs on right side.

Users can hover on the graph to view the metric value at the corresponding time. Zoom in/out are provided with the menu on top of each graph.

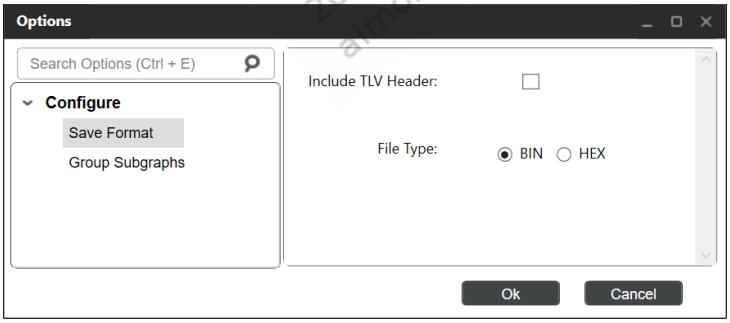


4.17 ALSALib exporter

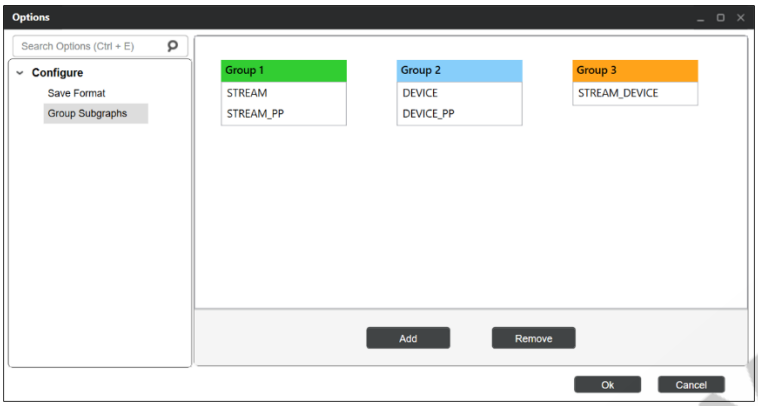
The ALSALib exporter feature supports the ability to export binary blobs to be consumed by the ALSALib plugin itself and/or the ADSP.

4.17.1 Configure options

Users can change the format of saved bin files in the **Configure -> Save Format** menu. It supports ability to include TLV header as well.



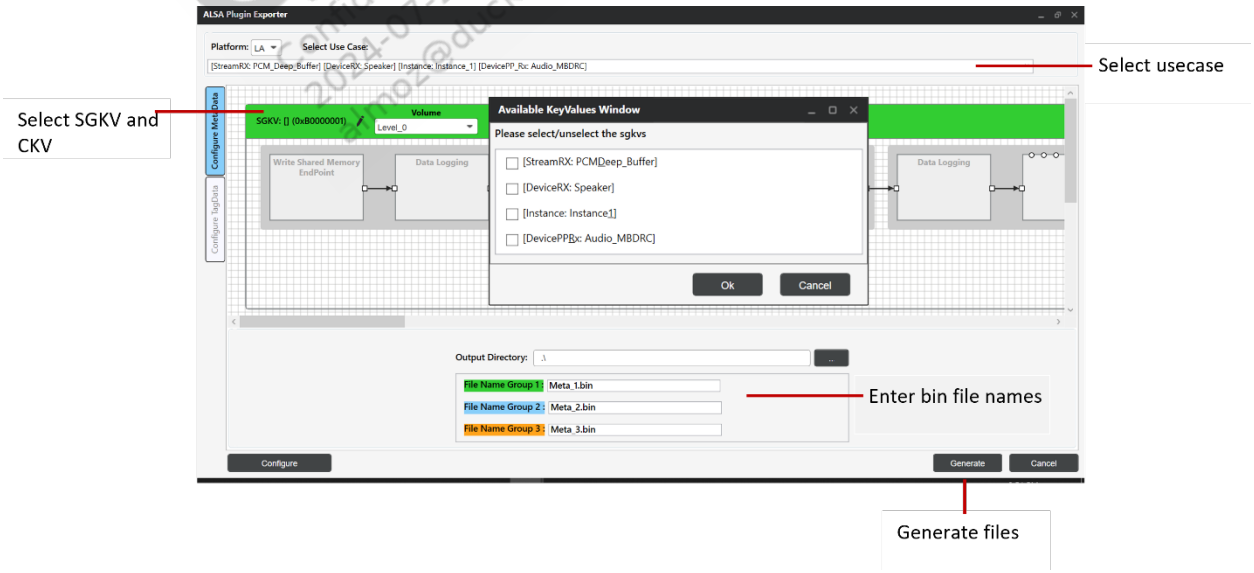
Users can also change the groups of subgraph types (if needed) using the **Configure -> Group Subgraphs** menu by dragging subgraph type from one group to another.



4.17.2 Configure and generate metadata

To configure and generate metadata:

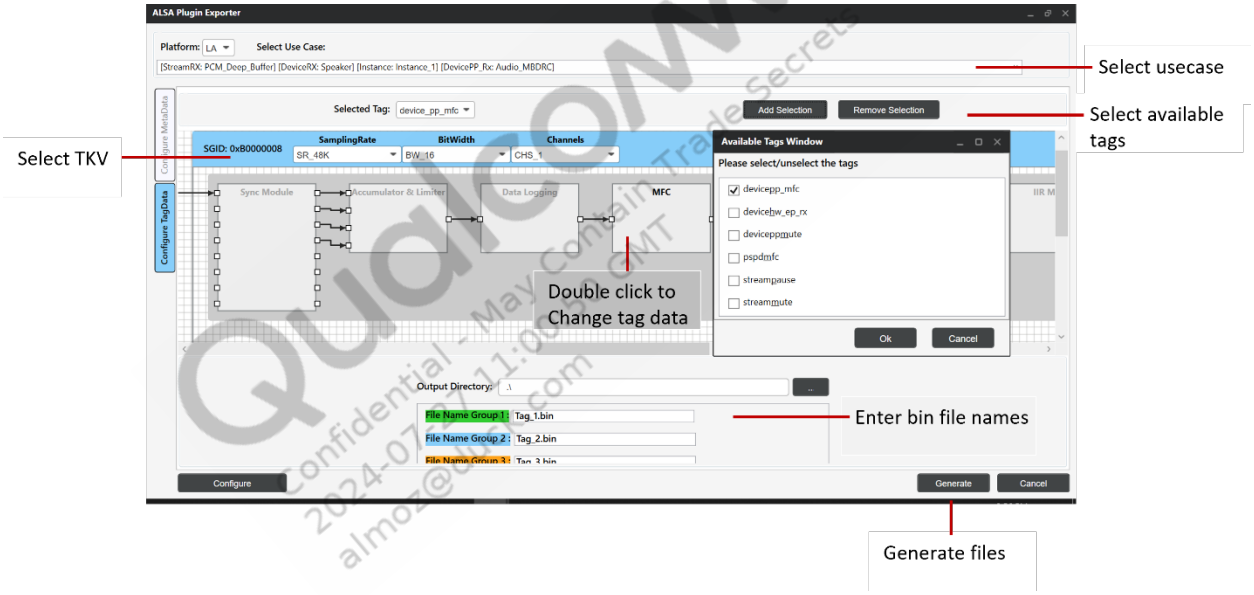
1. Click **Tools -> ALSALib Exporter**.
2. Select the Configure MetaData tab.
3. Select the use case for which metadata needs to be generated.
4. Select the CKV and edit the SGKV for each subgraph..
5. Enter the bin file names for each subgraph group.
6. Click **Generate** to create the bin files.



4.17.3 Configure and generate tag data

To configure and generate tag data:

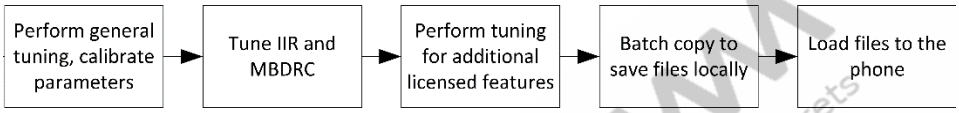
1. Click **Tools -> ALSALib Exporter**.
2. Select the Configure TagData tab.
3. Select the use case for which tag data needs to be generated.
4. Click **Add Selection** to select the tag for which data needs to be exported.
5. Select the TKV if applicable.
6. Double click on active modules to change tag data.
7. Enter the bin file names for each subgraph group.
8. Click **Generate** to create the bin files.



5 Tuning engineer

This role encompasses the responsibilities to tune the audio path to pass product expectations. The tuning engineer workflow occurs after the project configurator has made sure devices are in place and mapped. To better understand choices made by the project configurator, see Chapter 4.

The tuning engineer workflow is as follows:



5.1 Perform general tuning

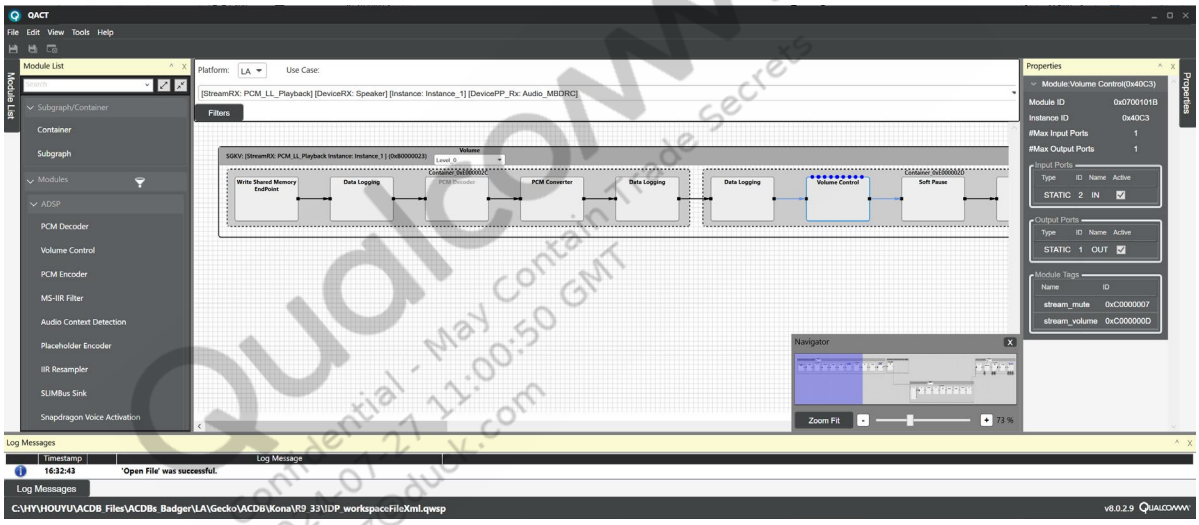
Three calibration modes are available:

- Non-Connected – Section 5.1.1
- Connected_Online – Section 5.1.2
- Connected_RTC – Section 5.1.3

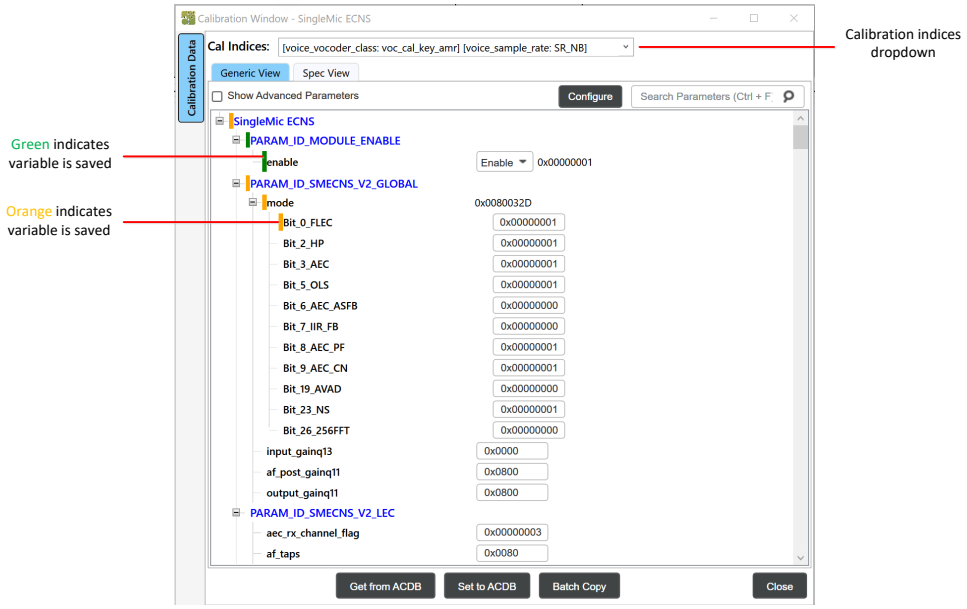
5.1.1 Non-Connected calibration

Non-Connected (Offline) calibration allows the user to open and modify calibration files without being connected to a target device. Settings saved in the modified calibration file can be pushed to a target device later. A power-cycle of the target device is then required to load the updated parameters into memory.

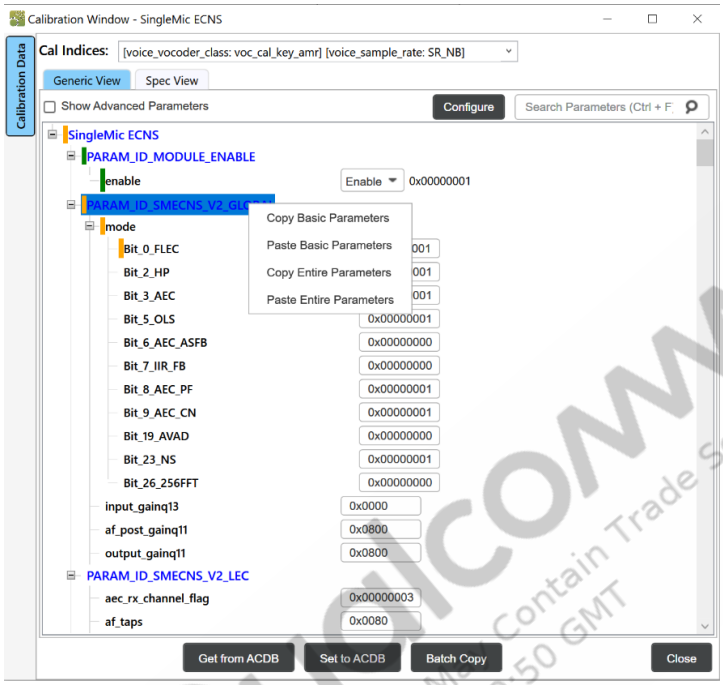
1. Click **Open File** on the home screen.
2. Select an .acdb file or .qwsp file and click **Open**.
3. The main window displays the selected graph(s) and parameters for the calibration file. A marked checkbox next to a parameter module name indicates that it has been enabled. Graph View is shown.



4. Double-click any module to display its parameters. In the following example, SMECNS has been selected.



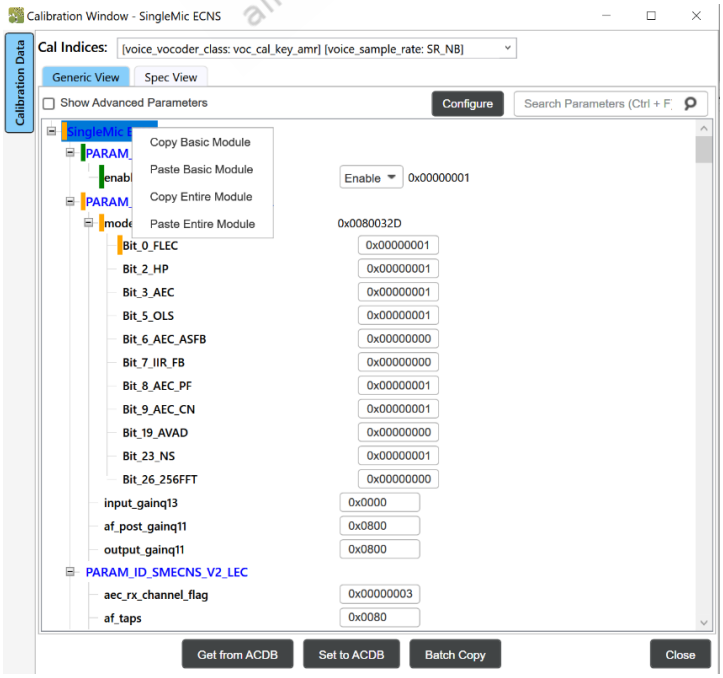
5. To modify individual values, enter new values in the applicable fields.
6. For many parameters, an array of values can be directly copied and pasted. Right-clicking a parameter name brings up a context menu from which to select.



For example, to copy the MODULE_ID_SMECNS_V2 values from one device use case to another:

a. Double-click the SMECNS module.

b. Right-click the root (Data) and select either **Copy Basic Module** or **Copy Entire Module**.



c. Close the SMECNS parameter window and select another use case from the Device use case drop-down.

d. Double-click the parameter module for which you would like to paste the copied values.

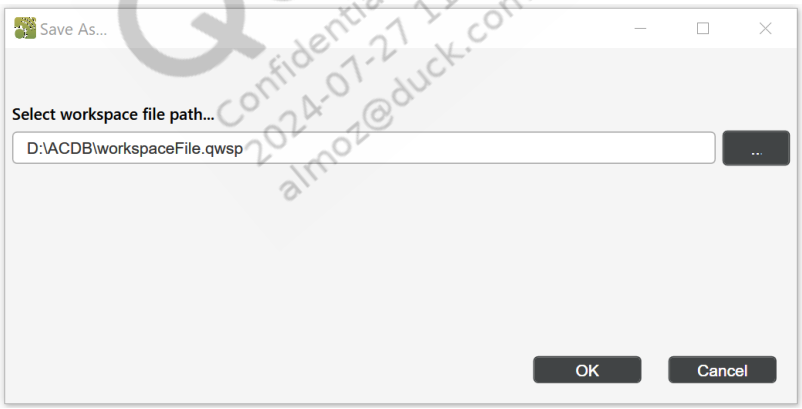
e. Right-click the root (Data) and select either **Paste Basic Module** or **Paste Entire Module**.

7. Click **Set to ACDB** to ensure that changes are saved to the .acdb file.
8. After all modifications have been completed, click **File**>**Save** or **File**>**Save as**.

5.1.2 Connected_Online calibration

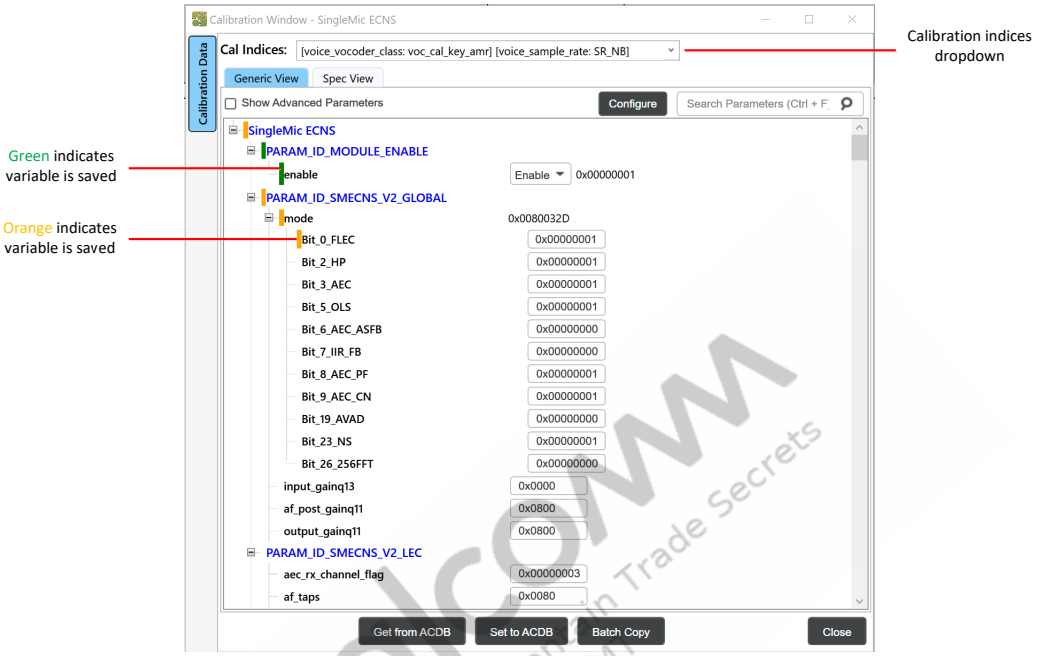
Connected_Online calibration copies the entire calibration file from the apps processor of the target device to a workstation. After the file has been updated, it is pushed to the target device. A device switch must occur on the target device in order to load the updated file.

1. Click **Connect to Phone** on the home screen. When the connection is successful, the status bar will indicate that the target is connected.
2. To save the calibration structure information in memory as an .acdb file:
   a. Click **File**>**Save as**.
   b. Click **Browse** to the right of the workspace path and navigate to the directory in which to save the calibration files. Click **OK**.



3. Select **View -> Connected DB Content**.
4. Select a GKV.
5. Double click a module.

6. Select the applicable parameters. If there is a checkbox, click the checkbox to enable the parameter. To modify individual parameter values, enter new values.

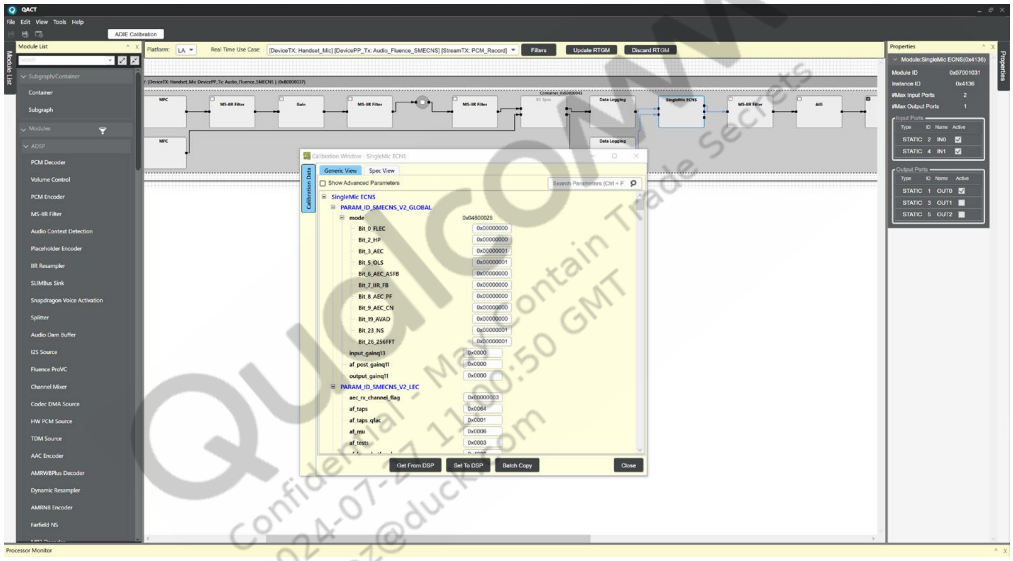


7. To save to the .acdb file, click **Set to ACDB**.
8. To retrieve the current value for a feature from memory, click **Get from ACDB**.
9. When all the modifications in the ACDB are complete, the entire calibration structure can then be pushed to the target. See Section 4.14.

5.1.3 Connected_RTC

Connected_RTC allows the user to change calibration data currently in use by the DSP of a connected target device. RTC does not provide access to the entire calibration file, it only accesses calibration data that is currently being used by the DSP. Therefore, when you perform RTC, it is *not* necessary to push updated calibration data to the target or force a device switch to load the updated calibration data. To perform RTC:

1. Ensure that a functional target device is connected to the workstation via USB:
   a. Click **Connect to Phone** on the home screen.
   b. Once a device is connected, QACT shows RTC calibration by default.
2. The use cases currently active in the DSP are shown.



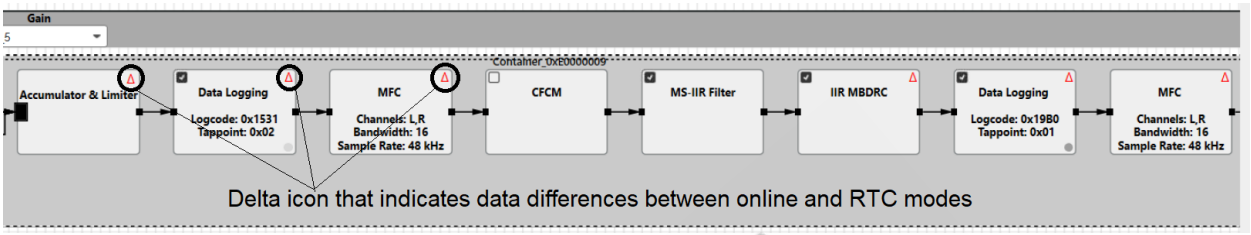
- If there are ongoing multiple RTC use cases, all of them will be shown in Real Time Use Case dropdown box. Select one or more to show them.
- The RTC calibration window for a module can be displayed by double-clicking the module.

NOTE: In RTC, it is necessary to activate the modules being tuned. For example, to tune the audio modules, it is necessary to activate the audio modules, and to tune voice, it is necessary to make a voice call.

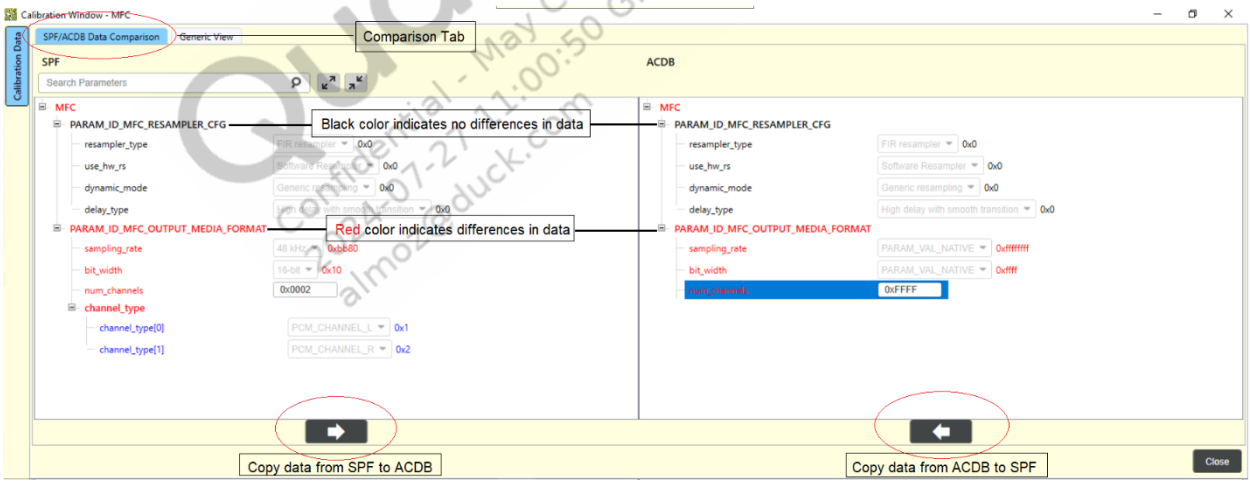
3. When DSP calibrations are finished, click **Set to DSP** to download your changes to the connected device.
4. To retain these DSP calibrations for future reference or to apply them to a different target, click **Batch Copy to ACDB**. This copies tuned data from RTC to Online mode so that the calibrated data in the DSP can be saved in an Online mode .acdb file. To save the calibration data as an .acdb file, click **File → Save as**.

5.1.4 Calibration data differences – Connected and RTC mode

In RTC Graph View, calibration data between connected and RTC modes is compared for each module. If the data is found to be different, a delta icon is added to the module block as shown in the following screenshot.



To see the exact differences in data between the payloads of parameters, navigate to the SPF/ACDB Data Comparison tab in the calibration window of the module. The data of parameters which can be set or tuned in both connected and RTC modes are shown in tree format. SPF data is shown in the left tree and ACDB data is shown in the right tree. Differences are highlighted in red color for easy identification. Data can be merged either from SPF to ACDB or vice-versa by clicking on the arrow buttons below the trees based on the desired direction of merge. This will copy the module data completely. Data is set to SPF/ACDB automatically.



NOTE: For certain modules such as MFC (which are tagged) or certain multi-channel modules such as MBDRC, Volume Control, etc., the calibration data can be overridden at runtime either by the HLOS or SPF. For such modules, the delta icon is shown on the module block in the graph view despite the data not being explicitly modified by the user to highlight the fact that there are data differences between connected and RTC modes.

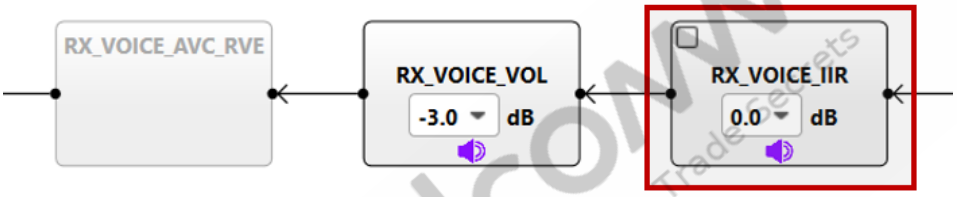
5.2 Tune IIR and MBDRC

5.2.1 Configure the IIR filter

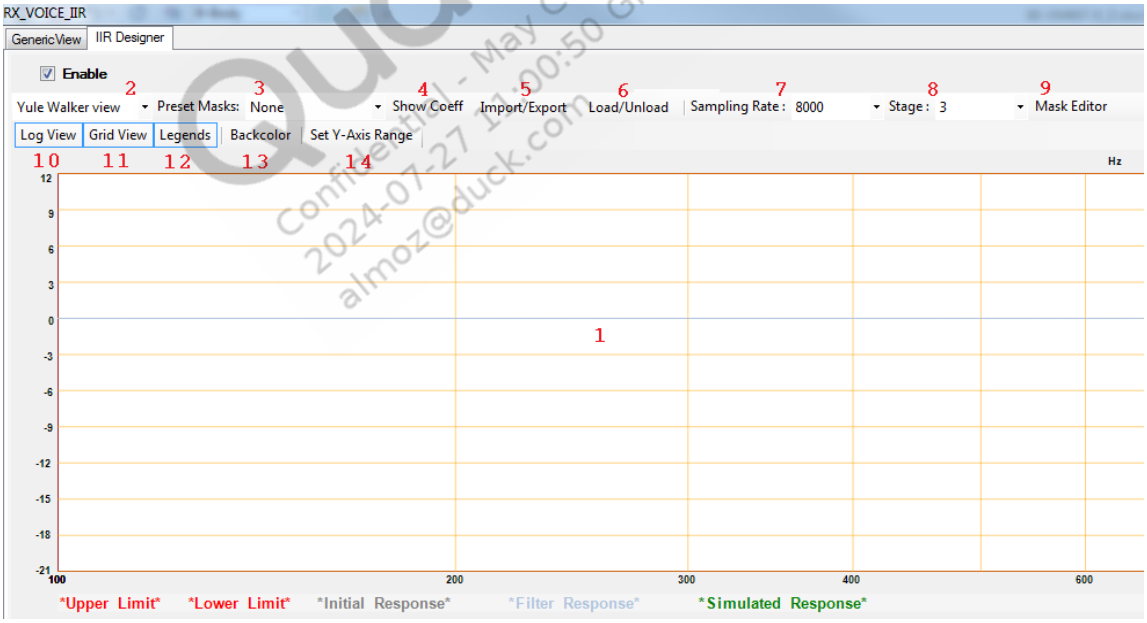
The IIR filter is configured using the IIR designer tool. This tool is a graphical filter design interface that allows the tuning engineer to design and save the filters required to achieve the intended frequency response characteristics.

5.2.1.1 Use IIR Designer

To launch the IIR designer, launch the IIR module on any path. For example, when tuning the voice Rx path, launch the IIR designer by double-clicking the RX_IIR module.



The top portion of the IIR designer window contains the filter output graph.

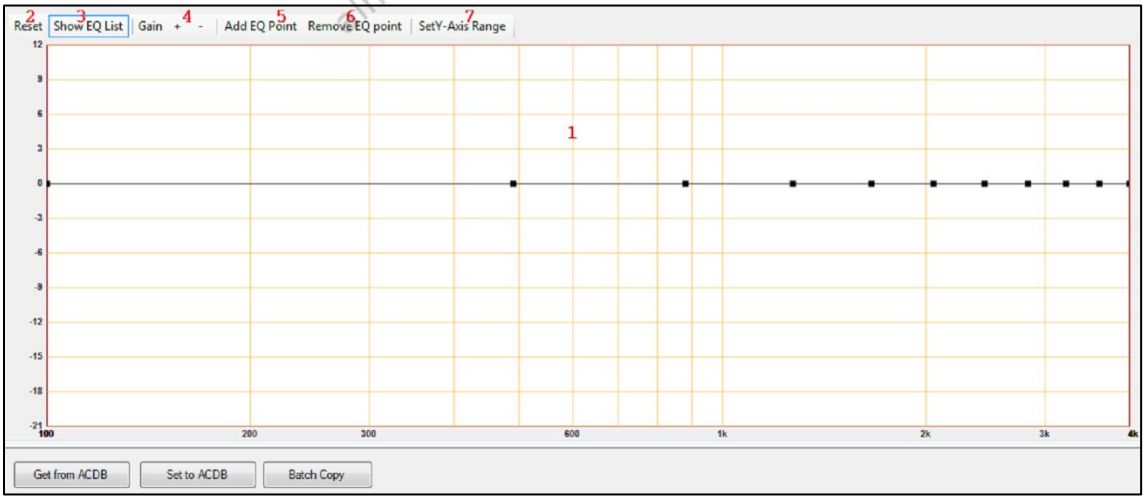


The following interface items correspond to the screenshot above.

1. Filter output graph – Shows the filter response, initial response (if loaded), simulated output response (if an initial response curve is loaded), and the tolerance mask (if loaded)
2. Yule Walker drop-down – Toggles between Yule Walker and Parametric filter design methods

3. Preset Masks – Allows the use of predefined frequency response tolerance masks available with the QACT installation
4. Show Coeff – Shows/hides IIR coefficients
5. Import/Export – Imports/exports coefficients or EQ points
6. Load/Unload – Loads/unloads an initial frequency response curve or custom frequency response tolerance mask
   - Initial frequency response curves may be obtained from test systems such as Head Acoustics ACQUA and Rhode&Schwarz UPV.
   - Custom frequency response tolerance masks may be designed as per OEM requirements.
7. Sampling Rate – Sets the sampling rate; in Online and Offline modes, this field is preselected; in RTC, the user must select the appropriate sampling rate manually
8. Stages – Sets the number of stages in the IIR filter; a maximum of 10 stages for voice, and 20 stages for audio record/playback are supported; this is not applicable to MSM6xxx and MSM7xxx chipsets
9. Mask Editor – Allows the tuning engineer to design custom tolerance masks; masks may be loaded using the Load/Unload button
10. Log View – Toggles the frequency axis between log scale and linear scale
11. Grid View – Shows/hides gridlines on the plot
12. Legend – Shows/hides the legend
13. Backcolor – Selects the color of the plot background
14. Set Y-Axis Range – Sets the Y-axis range

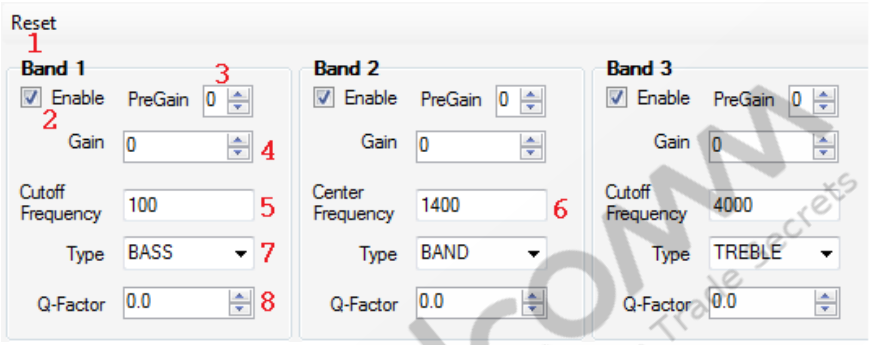
The lower portion of the IIR Designer window contains the EQ point design graph. This interface varies depending on whether Yule Walker or Parametric view is selected.



When Yule Walker view is selected, the following interface items are available and correspond to the numbers in the screenshot above.

1. EQ points graph – Allows for the editing of EQ points by dragging and dropping

2. Reset – Resets the filter
3. Show EQ list – Shows/hides the EQ points list; this list allows for finer control of the frequency and magnitude points
4. Gain +/- – Increases/decreases the gain of a selected EQ point
5. Add EQ Point – Adds additional EQ points for finer control of the filter
6. Remove EQ Point – Removes existing EQ points
7. Set Y-Axis Range – Adjusts the Y-axis range

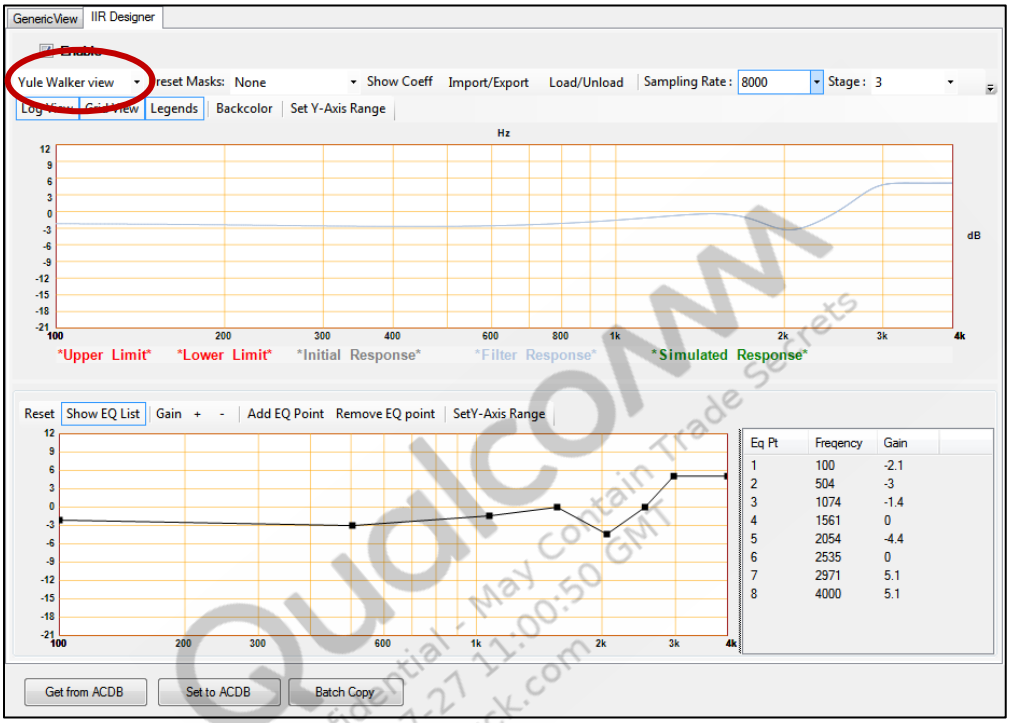


When Parametric view is selected, the following interface items are available and correspond to the numbers in the screenshot above.

1. Reset – Resets the filter
2. Enable – Enables the filter for the associated band; if unchecked, the band (stage) is bypassed
3. PreGain – Adjusts the amount of gain applied before the associated band
4. Gain – Adjusts the amount of gain in the filter design for the associated band
5. Cutoff Frequency – Sets the cutoff frequency for the bass and treble filter types
6. Center Frequency – Sets the center frequency for the band filter type
7. Type – Sets the filter type
   - Bass – Bass boost or bass cut shelving filter; gain is applied below the specified cutoff frequency
   - Band – Band boost or band cut filter; gain is applied in a band around the center frequency; the width of the band depends on the Q-factor selection
   - Treble – Treble boost or treble cut shelving filter; gain is applied above the specified cutoff frequency
8. Q-factor – Specifies the width of the band filter type; a higher Q-factor results in a sharper filter with smaller bandwidth

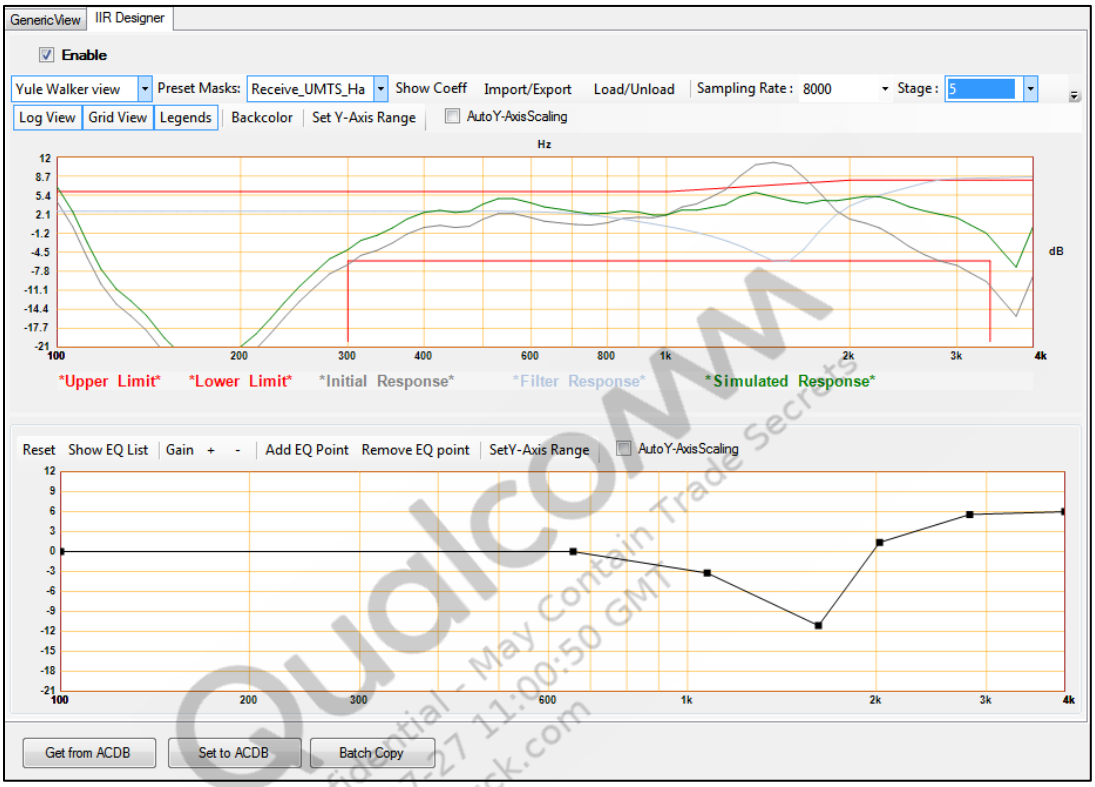
5.2.1.2 Design a filter using the Yulewalk method

The Yulewalk method allows the tuning engineer to draw a curve by clicking and dragging EQ points on the bottom graph. See http://www.mathworks.com/help/signal/ref/yulewalk.html for details about this method. The Yulewalk method is limited by low frequency resolution at higher sampling rates.



1. Double-click the IIR module to open IIR Designer.
2. Ensure that Yulewalker view is selected.
3. If in RTC mode, select the appropriate sampling rate.
4. Drag the marked points in the bottom graph to achieve the intended filter response.
5. To guide the process of designing the filter, an initial response curve and frequency response mask can be imported.
   a. To load an initial response curve, click **Load/Unload** and select **Load Initial Response**.
   b. To load a preset mask, click **Preset Masks** and select an appropriate mask.
   c. To load a custom mask, click **Load/Unload** and click **Load Mask**.
   d. To design a custom mask, click **Mask Editor**. Design and save a mask as required, then load the mask as described in Step c.

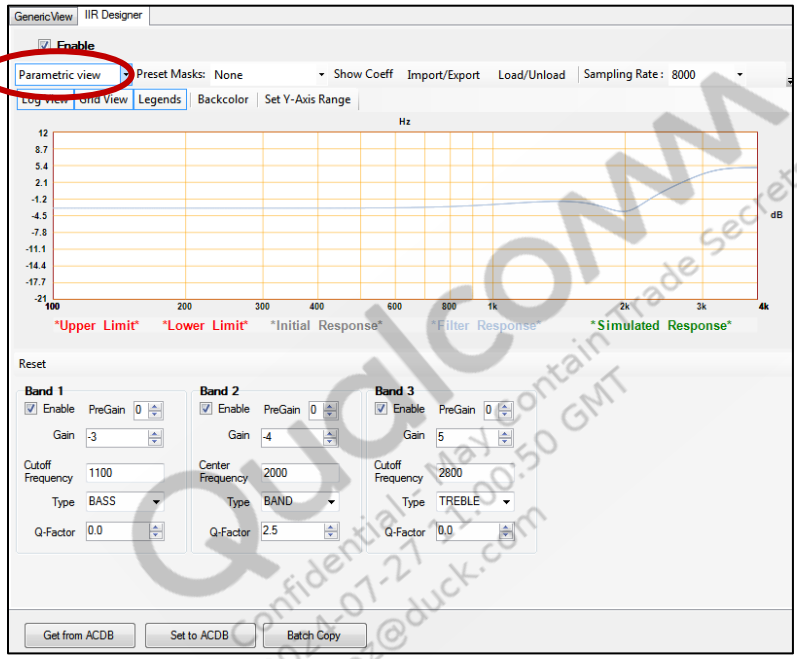
The following example plot illustrates the case where the steps in this procedure have been performed. The green curve is the simulated filter response which should match the frequency response measurements on a test system such as Head Acoustics ACQUA.



5.2.1.3 Design a filter using Parametric Designer

The Parametric Designer provides a parametric equalizer interface to design filters. It consists of 10 bands, each able to be configured as a bass, treble, or band filter. The bass and treble filters are shelving filters, while the band filter is a bandpass/bandcut filter. The band filter also has a quality factor (Q-factor) that allows the user to adjust the sharpness of the band filter.

1. Double-click the IIR module to open IIR Designer.
2. Click **Yule Walker view** and change the selection to Parametric view.



3. If in RTC mode, select the appropriate sampling rate.
4. Starting with band 1, set the filter type to bass, treble, or band using the Type field.
5. Set the cutoff/center frequency.
6. Set the gain value to the required amount of equalization using the Gain field.
7. If the band is of the band filter type, adjust the Q-factor to determine the sharpness of the filter using the Q-Factor field.
8. If further correction is needed, repeat Steps 4 to 7 to add more bands.
9. To guide the process of designing the filter, an initial response curve and frequency response mask can be imported:
   a. To load an initial response curve, click **Load/Unload** and select **Load Initial Response**.
   b. To load a preset mask, click **Preset Masks** and select an appropriate mask.
   c. To load a custom mask, click **Load/Unload** and click **Load Mask**.
   d. To design a custom mask, click **Mask Editor**. Design and save a mask as required, then load the mask as described in Step c.

5.2.1.4 Save the created filter

1. Click Import/Export.
2. Select Export EQ Points.
3. Navigate to the location in which to save the points.
4. Click Save.

To save the calibration data for the IIR module:

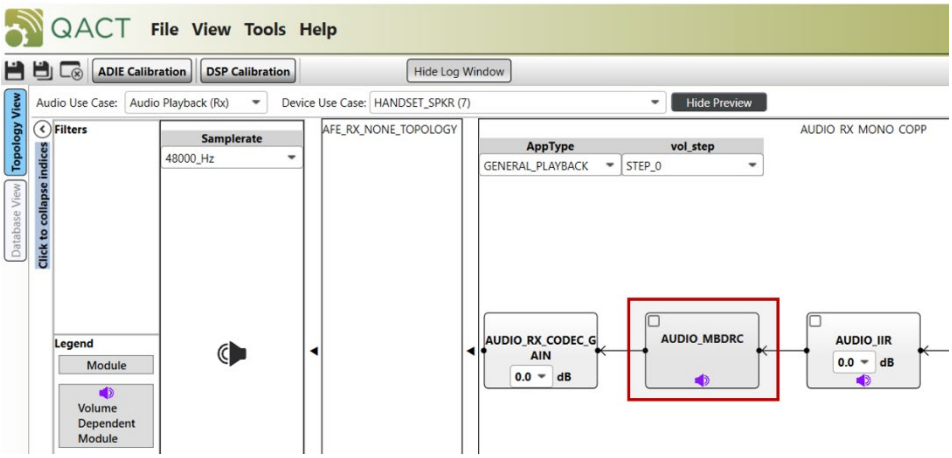
- Offline mode – Click **Set to ACDB**. To copy the filter to other applicable use cases, click **Batch Copy** and select the destination indices in which to copy the filter.
- Online mode – Click **Set to ACDB** to save the filter to the database on the target. To copy the filter to other applicable use cases, click **Batch Copy** and select the destination indices in which to copy the filter. Click **File>Save As** and save the new calibration data. Follow the workflow to update the device with the new calibration data.
- RTC mode – Click **Set to DSP** to save the filter to the DSP. This calibration data will be lost if the use case, e.g., an ongoing voice call, is ended. To prevent data loss, click **Batch Copy** and copy back to the active use case. This will save the calibration to the database on the device. Follow the procedure for Online mode to persistently save the calibration data.

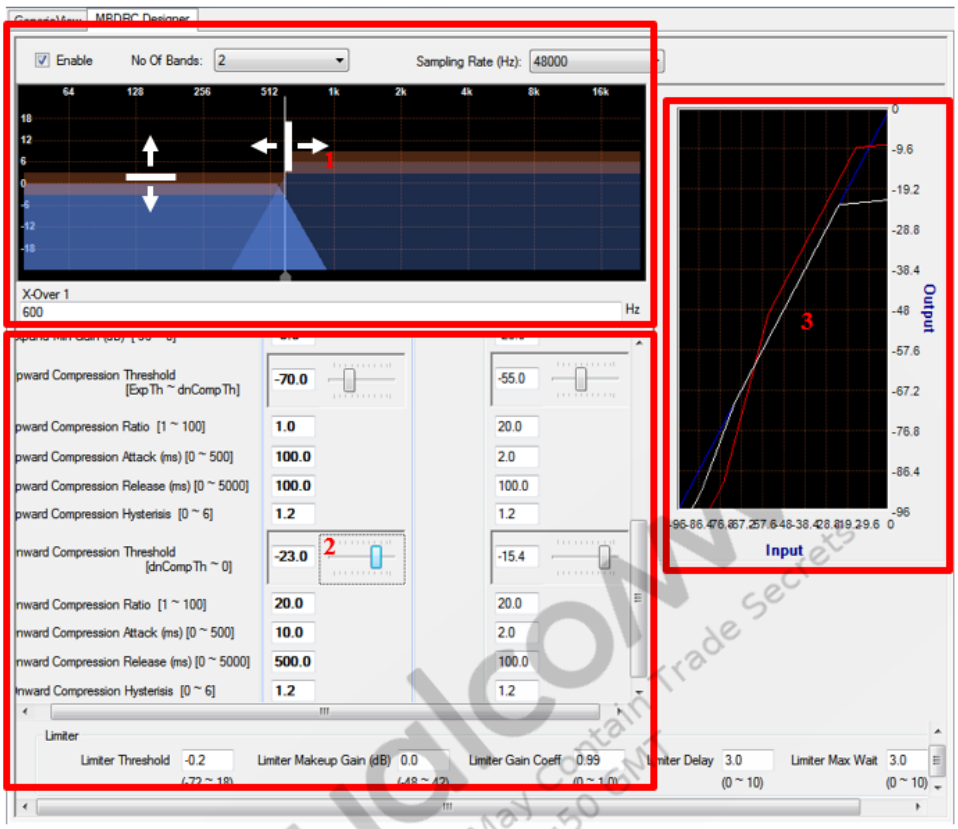
5.2.2 Configure MBDRC

MBDRC is configured using the MBDRC Designer tool. This tool is a specialized design interface that allows the tuning engineer to easily tune the MBDRC module.

5.2.2.1 Use MBDRC Designer

To launch the MBDRC designer, launch the MBDRC module on any path, for example, when tuning the voice Rx path, launch the MBDRC designer by double-clicking the RX_MBDRC module.



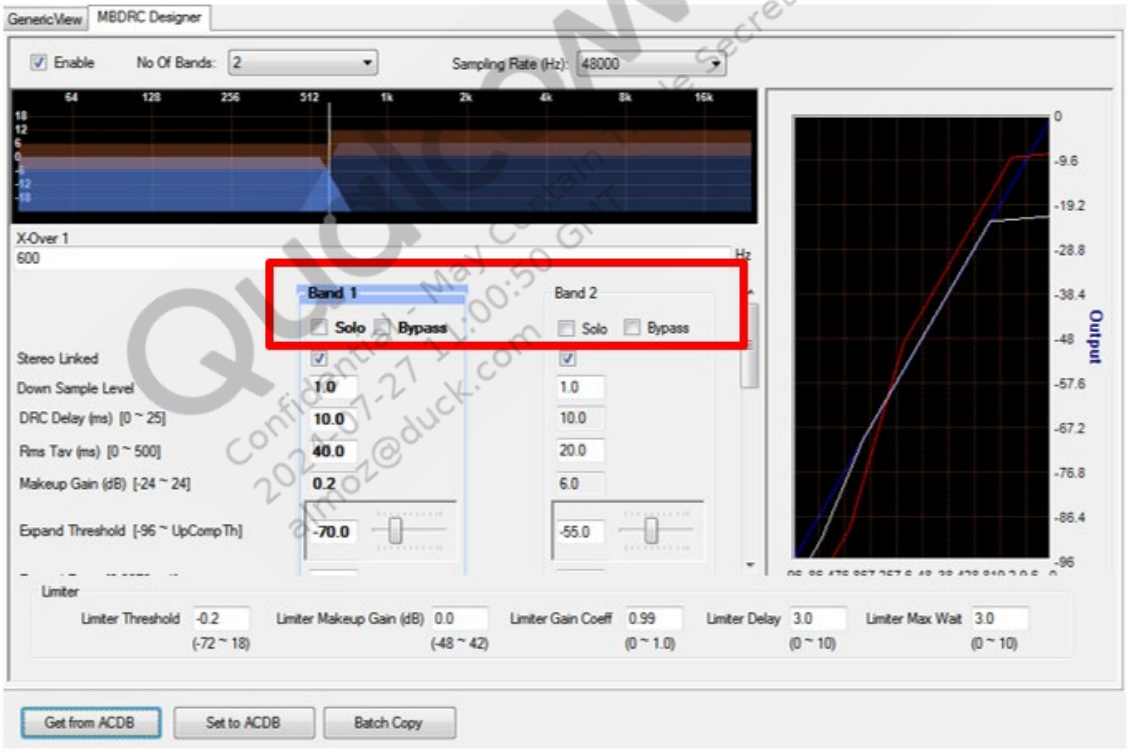


The MBDRC designer interface consists of the following three sections (items correspond to the numbers in the screenshot above).

1. MBDRC configuration – This area is used to configure the MBDRC settings that are required before starting DRC fine tuning for each band. The following can be adjusted from the MBDRC configuration box:
   - Enable – Enable/disable MBDRC
   - No. of Bands – Sets the required number of bands of DRC
   - Sampling Rate – Sets the sampling rate; in Online and Offline modes, this field is preselected; in RTC, the user must select the appropriate sampling rate manually
   - X-Over Frequencies – Crossover frequencies may be manually entered in the text box or adjusted graphically by moving the vertical cursor to the left or right
   - Makeup Gain – Makeup gain can be adjusted graphically by moving the horizontal cursor up or down
2. Parameter tuning box – This area lists the DRC parameters in native units (such as milliseconds, decibels, etc.). Edit the fields to adjust parameter values.
3. Static level curve – This area shows the designed static level curves for each DRC band. This curve is a graph of output level vs input level and reflects the settings for the various compression and expansion thresholds as well as the makeup gain. The active band selection that is being edited, or was last edited, is shown in white, while other bands appear in red. The blue line represents a linear curve where no compression or expansion happens.

5.2.2.2 Configure MBDRC

1. Enable MBDRC.
2. Set the desired number of bands. This can be adjusted later when fine-tuning.
3. Set the X-over frequencies. Make note of the X-over frequency values for future use.
4. Select Solo or Bypass for each band as appropriate:
   - Solo – Mutes all other bands; if more than one band has Solo enabled, then all bands with Solo enabled are active and others are muted
   - Bypass – Bypasses the DRC processing for that band; the makeup gain remains active for that band.
5. Adjust the DRC parameters for each band to achieve intended static level curves.
6. Fine-tune the limiter parameters as needed.



5.2.2.3 Save the calibration data for MBDRC

To save all calibration data for MBDRC (except X-over frequencies):

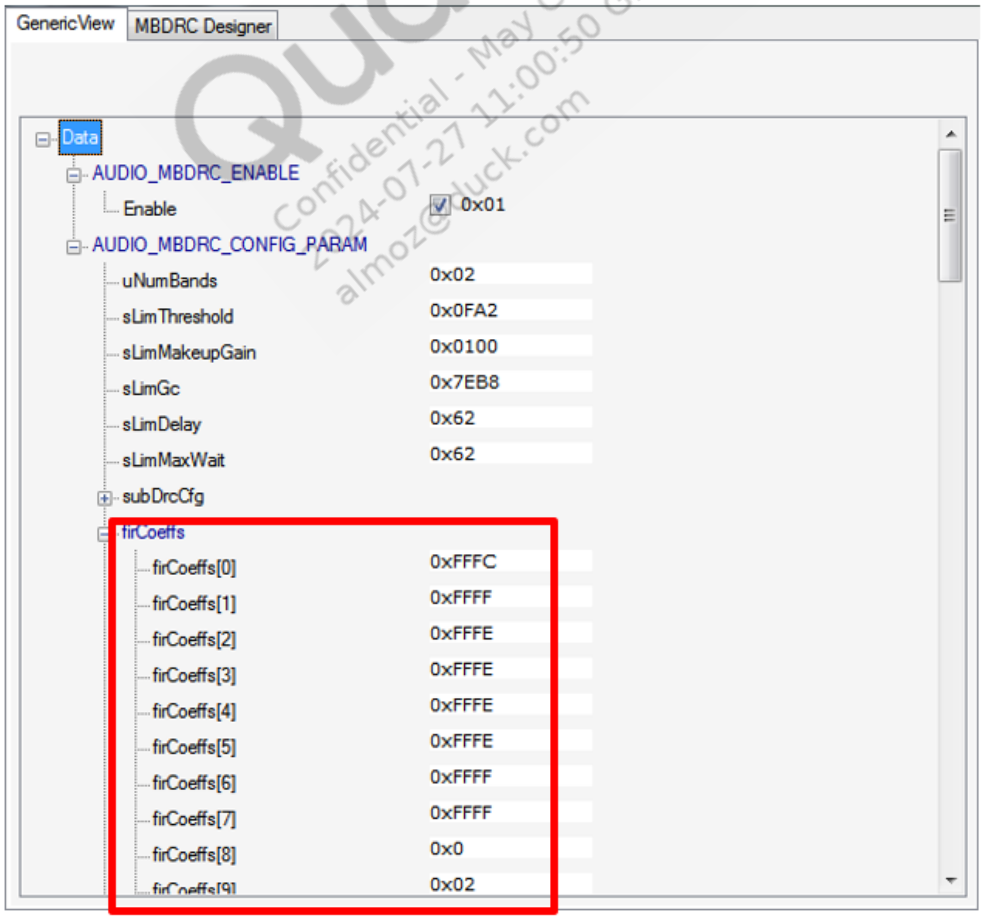
- Offline mode – Click **Set to ACDB**. To copy the filter to other applicable use cases, click **Batch Copy** and select the destination indices in which to copy the filter.

- Online mode – Click **Set to ACDB** to save the filter to the database on the target. To copy the filter to other applicable use cases, click **Batch Copy** and select the destination indices in which to copy the filter. Click **File>Save As** and save the new calibration data. Follow the workflow to update the device with the new calibration data.
- RTC mode – Click **Set to DSP** to save the filter to the DSP. This calibration data will be lost if the use case, for example, an ongoing voice call, is ended. To prevent data loss, click **Batch Copy** and copy back to the active use case. This will save the calibration to the database on the device. Follow the procedure for Online mode to persistently save the calibration data.

5.2.2.4 Save X-over frequencies

MBDRC v1 and v2

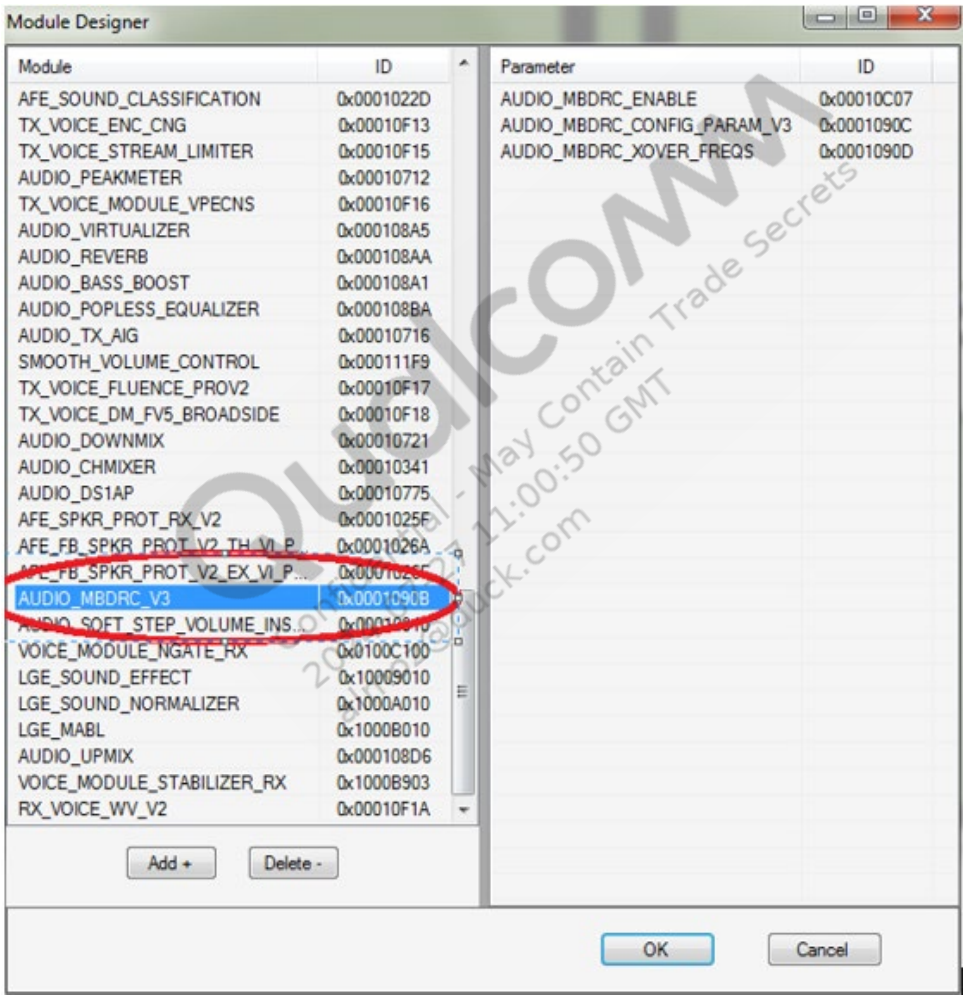
X-over frequency adjustment is a feature of the tool and is not part of the calibration data structure. As a result, any tuned values will not persist beyond one session. This does not mean that the tuning is affected; filter coefficients are calculated from the X-over frequencies and saved in the calibration data as seen in the Generic View tab under firCoeffs. It is recommended to manually record the X-over frequency value after tuning MBDRC v1 and v2 so that same value can be used for future tuning.



If a new session is opened, the X-over frequencies are reset to the default values. To further adjust the MBDRC parameters at this point, restore the X-over frequency display to the original setting manually and adjust any parameters required.

MBDRC v3

MBDRC v3 allows the persistence of X-over frequencies. To check if MBDRC_V3 is available in a given build, open the Module Designer and check if the AUDIO_MBDRC_V3 module is defined in the .acdb file. If it is, this feature may be utilized by creating a topology that uses AUDIO_MBDRC_V3.



5.3 Tune additional licensed features

5.3.1 Speaker Protection

Demand for high audio volume can push speakers beyond safe operating limits to the point of device damage. To ensure the safe operation of mobile device speakers without impacting the user listening experience, the Speaker Protection feature integrates a voltage and current sense to monitor the mobile device speaker transducer in real-time and send feedback signals to the DSP.

The Feedback Speaker Protection v2.0 feature includes:

- Stereo speaker protection
- Diaphragm excursion monitoring and control
- Overtemperature monitoring and control
- Analog clip monitoring and control

The Speaker Protection tuning tool allows the user to observe the performance of the Speaker Protection algorithm by monitoring algorithm variables sent in the form of diag log packets. Analysis can be performed on the data received in the tool to generate the coefficients used in the feedback speaker protection algorithm. To launch the Speaker Protection tuning tools, click **Tools > Tuning>Feedback Speaker Protection** or **Tools > Tuning> Speaker Protection v2 Tuning Tool**. See *Application Note: Voltage-Current Feedback Speaker Protection V2.0 and Tuning Process* (80-NT796-1) for more information.

5.3.2 ANC/AANC

ANC directly cancels out unwanted background noise at the user's ear by projecting antinoise through a speaker or receiver. This allows the user to experience a quiet environment even when there is considerable background noise.

ANC technology is based on specially designed digital hardware integrated with a codec. This design achieves a minimal processing delay of < 40 µs to guarantee high performance. ANC also has the generalization capabilities to support many applications and codec use cases. Applications supported are:

- Headset Feed Forward (FF) ANC – This takes the reference noise before the noise arrives at the actuator, computes an inverse form of the noise, and plays the antinoise through the speaker. FF ANC is simple, effective, and can be used with virtually all acoustic designs, e.g., ear inserts, ear cups, or earpieces. Small ear insert-type devices or earpiece devices generally use FF ANC because it is difficult to integrate a mic after the speaker. However, without careful design, an FF system can suffer directional performance variation when the noise source direction is not fixed in the 3D space.

- Headset Feedback (FB) ANC – This uses a mic to capture noises between the ear canal and the speaker. FB ANC is more complicated than FF ANC since the noise mic also picks up music and voice signals sent to the speaker. Without sophisticated control, this acoustic leakage can lead to distortion. FB ANC is typically more effective at low-frequency noise reduction and has less directional performance variation than FF ANC.
- Handset Adaptive ANC (AANC) – New for version 2, AANC is a Digital Signal Processor (DSP) firmware that uses signals from two ANC mics to adaptively control FF ANC in the codec hardware. For earpiece ANC application, AANC can provide robustness against holding position and pressure. AANC is highly integrated with the audio codec and supports a wide variety of codec operating frequencies. This integrated design allows an external processor to access the ANC mic input signals through the codec PCM interface.

The ANC tuning tool allows the user to tune the ANC feature and generate DSP AANC parameters. P-path, S-path, and E-path recordings are collected, converted from 48 kHz to 8 kHz/16 kHz, and analyzed. From these recordings, ADIE ANC and DSP AANC parameters are generated. AANC parameters can be monitored in RTC/RTM mode and calibrated in real time simultaneously. To launch the ANC tuning tool, click **Tools > Tuning>ANC Tuning Tool**. See *ANC Tuning Guide* (80-NK803-1) for more information.

5.3.3 Fluence

Fluence v5 is Qualcomm's proprietary echo cancellation and noise suppression solution. It supports Single Microphone, Dual-Mic Endfire, and Dual-Mic Broadside modes.

The Fluence Tuning Tool (FTT) allows the user to observe the performance of the Fluence algorithm by monitoring the algorithm variables sent to the PC as diag log packets. Recordings can be saved onto the PC and analyzed offline using Offline mode functionality. To launch the FTT, click **Tools > Tuning>Fluence Tuning Tool – Offline Mode**.

FTT supports the following module IDs:

- TX_VOICE_FV5ECNS_SM
- TX_VOICE_FV5ECNS_DM
- TX_VOICE_DM_FV5_BROADSIDE

See the following documents for more information:

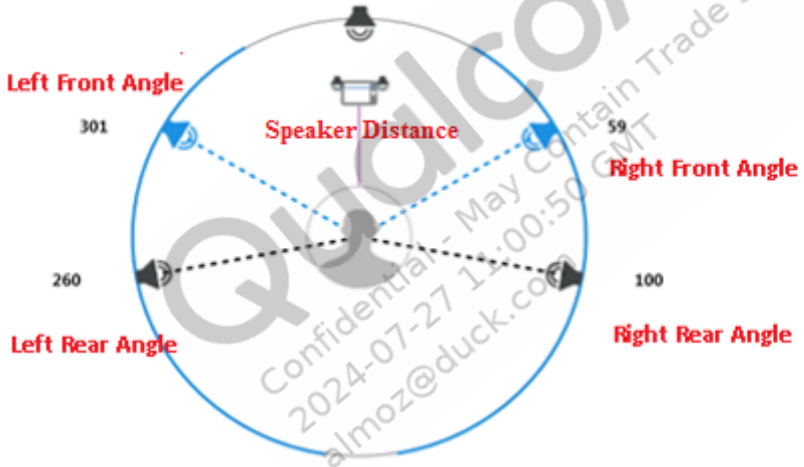
- *Presentation: Fluence v5 Acoustic Echo Cancelation Audio Tuning Training* (80-NK880-2)
- *Presentation: Fluence v5 Noise Suppression Audio Tuning Training* (80-NK880-3)
- *Presentation: Fluence v5.5 Broadside Noise Suppression Audio Tuning Training* (80-NK880-4)

5.3.4 AudioSphere

AudioSphere is a solution for free-field surround sound. QACT provides a GUI interface called the AudioSphere Visual Designer for tuning this feature.

To tune AudioSphere:

1. Double click the AudioSphere module to open the AudioSphere Visual Designer.
2. Select one of the following tabs:
   - Stereo Speaker Positioning –This tab is used to apply the AudioSphere effect to stereo playback content
   - 5.1/7.1 Speaker Positioning – This tab is used to apply the AudioSphere effect to multichannel content
3. Set the values for the front and rear angles by selecting the front and rear speakers and moving them using the mouse or up/down keys. The highlighted part on the ellipse shows the bounds after which the speakers will not move.



4. In the Base Parameters text fields, configure the distance between the speakers on the headset and set the base distance between the headset and the user's head. As the parameters are updated, the corresponding highlights are displayed in the GUI.
5. When configuration is complete, Click **Set to ACDB** to save the AudioSphere calibration data. See Section 5.4 for instructions on using the Batch Copy function to copy the calibration data to other destination indices.

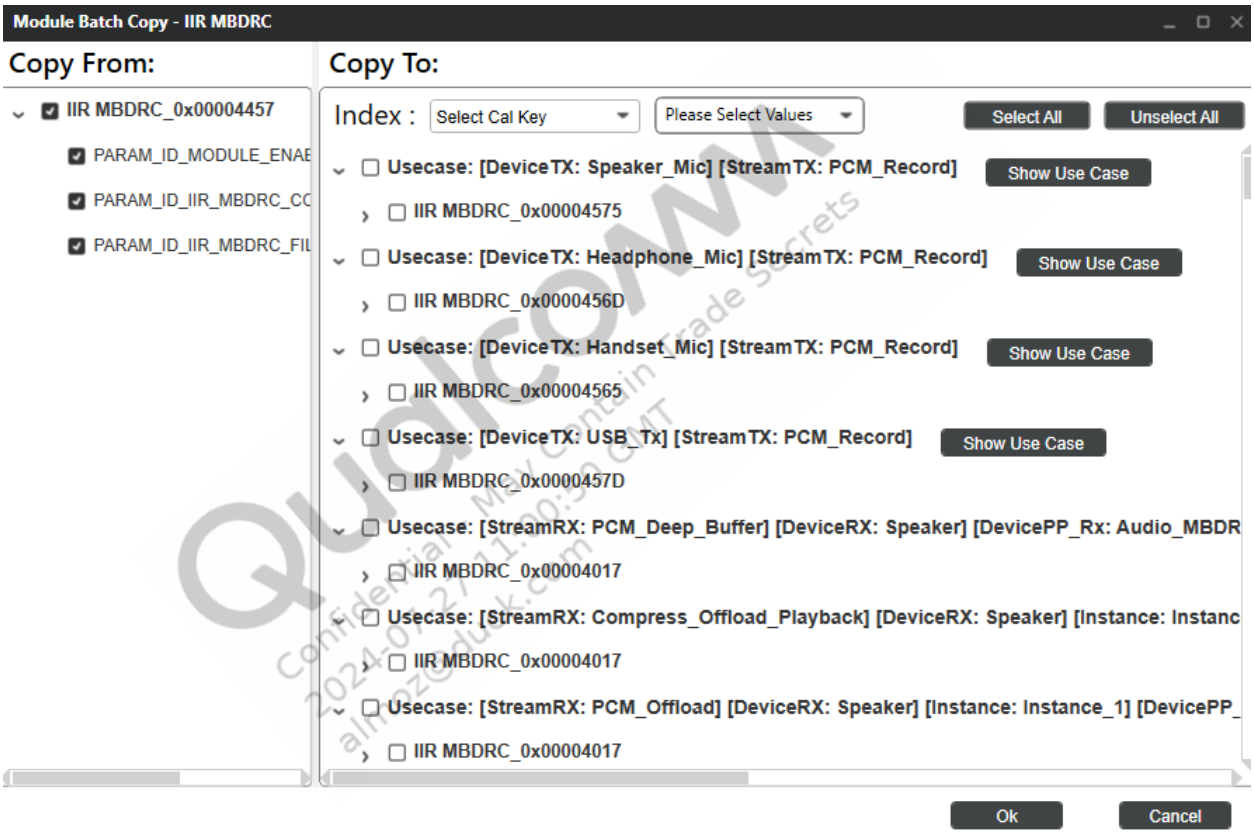
5.4 Batch copy to save files locally

Two types of batch copy is supported.

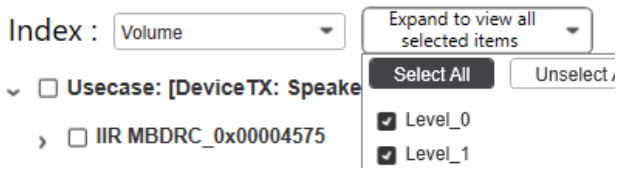
1. Module batch copy
2. Use case batch copy

5.4.1 Perform module batch copy

1. Double-click a module and, if necessary, make appropriate adjustments.
2. Click **Batch Copy**. The Batch Copy form displays.
3. In the COPY FROM pane, select the parameters from which to copy.
4. In the COPY TO pane, select the appropriate combinations.
5. To select all use cases, click on the “Select All” button at the top.



6. Users can also search for and select specific keys.



7. Click **OK**. A message similar to the following will appear in the log window to indicate the batch copy result.

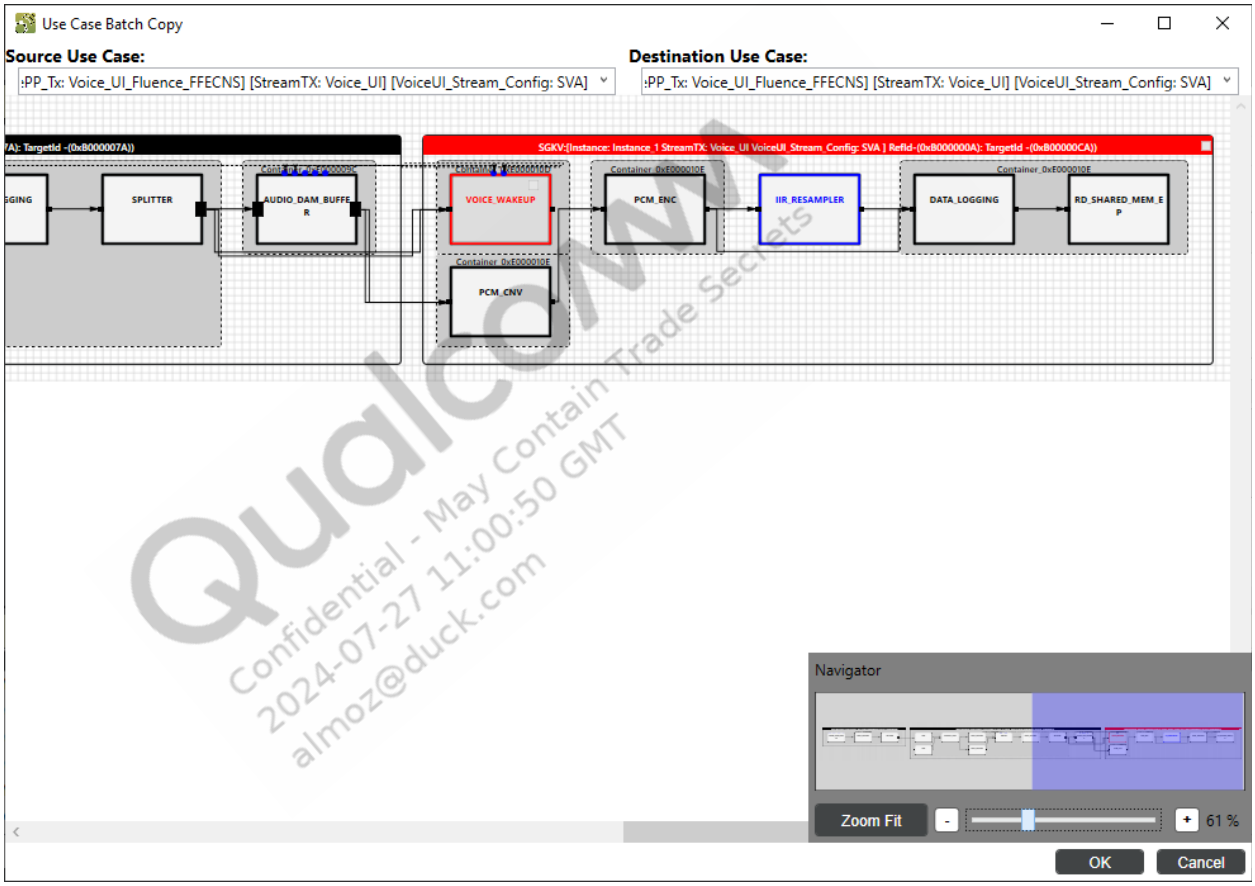
```
Started Cal Data Batchcopy ....
...Completed Cal Data Batchcopy.
```

8. Click **File>Save as** and navigate to the directory in which to save the file.

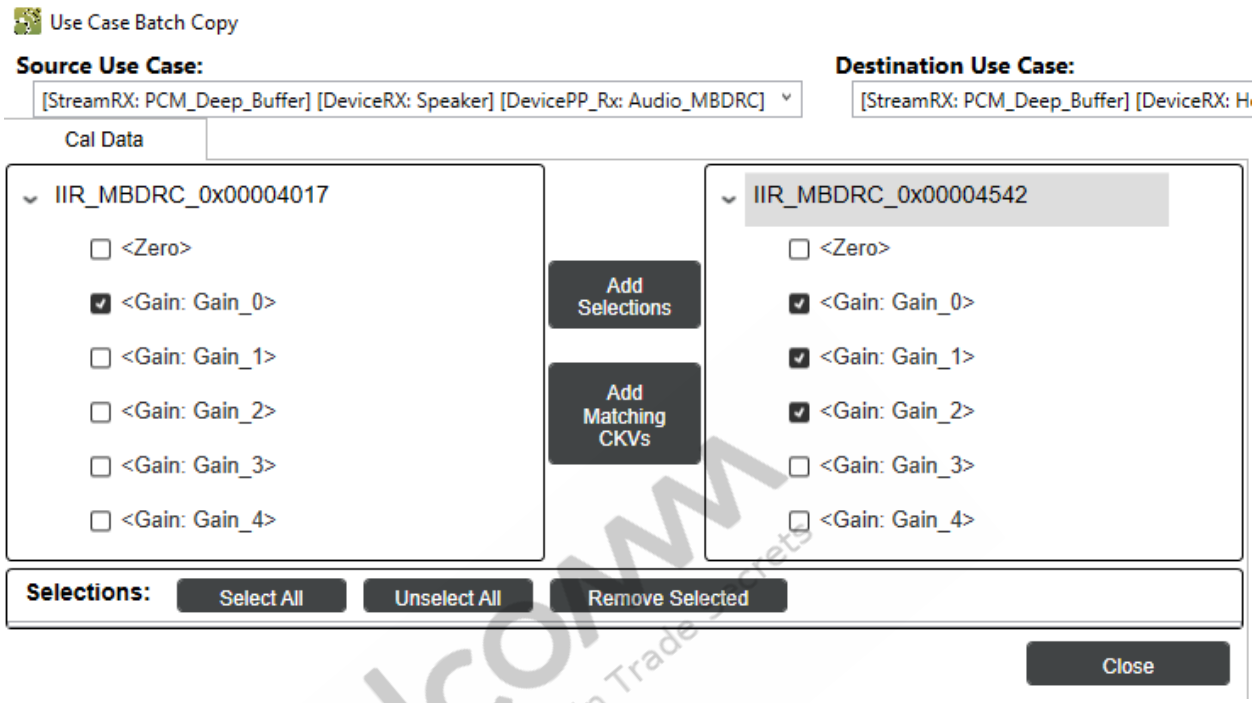
9. Click **OK**.

5.4.2 Perform use case batch copy

1. Click **Use Case Batch Copy**. The Batch Copy form displays.
2. Select the **Source Use Case** from the left pane and **Destination Use Case** from the right pane. Source use case will contain the list of use cases selected in the Graph View.



3. Once comparison is complete, the graph shows the same/modified/new/deleted modules.
   - **Black** color denotes the module data is same between both use cases.
   - **Red** color denotes that module data is different between both use cases.
   - **Blue** color denotes that module is present in source and not present in destination use case.
   - **Grey** color denotes that module is present in destination but not in source use case.
4. Click on module(s) with **Red** color to copy data from source to destination.



5. Select cal keys on left and right and click **Add Selections**.
6. Click **Close**.
7. Users can also select matching CKVs to copy from Gain_0 to Gain_0, Gain_1 to Gain_1 etc.
8. In case module has differences in both Cal Data and Tag Data, two tabs will be shown to the user.
9. Click OK to copy for all the selected modules.

5.5 Load files to a target device

5.5.1 LA

NOTE: This process may change from chipset to chipset. Users are encouraged to contact QTI support.

1. To load ACDB files to an LA target device, execute the following commands in the adb shell:

```
adb root
adb remount
adb shell "rm -r etc/acdbdata"
```

2. Copy all .acdb files from vendor\qcom\proprietary\mm-audio\audcal\family-b\acdbdata\
<target>\<form factor> to the local folder, say c:\temp.
3. Open a PC command prompt and go to c:\temp.
4. Push all .acdb files to the folder used on the device, e.g.:

```
adb push MTP_Bluetooth_cal.acdb /etc/.
adb push MTP_General_cal.acdb /etc/.
adb push MTP_Global_cal.acdb /etc/.
adb push MTP_Handset_cal.acdb /etc/.
adb push MTP_Hdmi_cal.acdb /etc/.
adb push MTP_Headset_cal.acdb /etc/.
adb push MTP_Speaker_cal.acdb /etc/.
```

5. `adb shell sync`
6. Reset the phone or restart mediaserver for new .acdb files to take effect.

   As an alternative, instead of restarting the device you can kill the mediaserver process via the following commands:

```
adb shell
ps "mediaserver"
kill -9 <mediaserver pid>
```

   This should restart the mediaserver process which reinitializes the ACDB.
7. If everything was done correctly, you should be able to see valid sizes for all 7 .acdb files. When the files are displayed, run the following commands:

```
adb root
ls -l /etc/*.acdb
```

5.5.2 QNX

Use the instructions in this section to load audio_cal.acdb to EFS for MSM8974.

To load .acdb files that were created in Offline mode to the file system on the target device via file copy and device switch:

1. Navigate to the directory on the QACT workstation where the appropriate .acdb files are stored.
2. On the target device, search for *.acdb file(s) to determine its location on the target device.
3. On the QACT workstation, copy the modified *.acdb file(s).
4. On the target device, paste and overwrite the existing *.acdb file(s).
5. Reboot the device. The ACDB software will search for the *.acdb file(s) and store the delta of the .acdb file(s) under the file system and built-in database.

5.5.3 WA and WP

Use the instructions in this section to load *.acdb file(s) to EFS for MSM8960 WA and WP targets.

To load .acdb files that were created in Offline mode to the file system on the target device via file copy and device switch:

1. Navigate to the directory on the QACT workstation where the appropriate .acdb file is stored. This folder is found in the path from where the ACDB driver is installed.
2. On the target device, search for *.acdb file(s) to determine its location on the target device.
   - For WA, this will be a directory named Liquid in the path from where the ACDB driver is installed.
   - For WP, this will be a directory named Fluid in the path from where the ACDB driver is installed.
3. On the QACT workstation, copy the modified .acdb file(s).
4. On the target device, paste and overwrite the existing .acdb file(s).
5. Reboot the device. The ACDB software will search for the .acdb file(s) and store the delta of the .acdb file(s) under the file system and built-in database.

Part 3: WCD/WSA EVB Products

6 Using QACT with WCD/WSA EVB products

For WCD/WSA EVB Product(s), there are no QWSP/ACDB files to open.

- After connecting to a WCD Evaluation Board, the Engineering View is shown to allow for execution of QCRG scripts and debugging of the hardware interface.
- After connecting to a WSA Evaluation Board, the WSA Evaluation View is shown to provide an audio player with QTI signal processing simulation for the evaluation of WSA and Speaker Protection performance.

6.1 WCD Evaluation Board – Engineering View

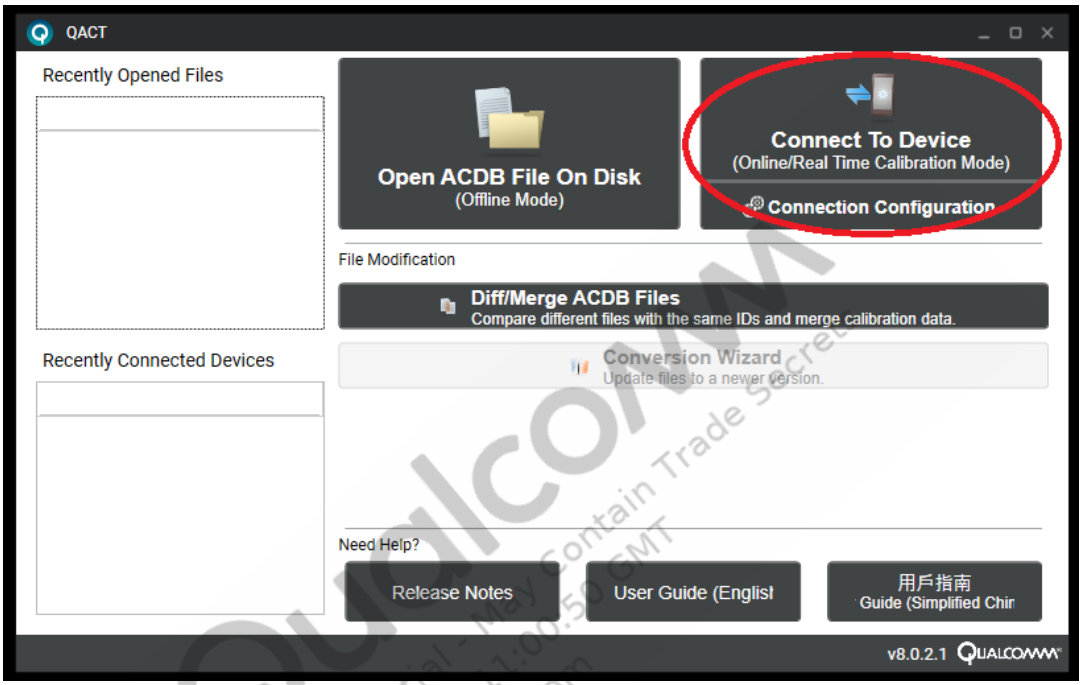
Refer to Sections 4 through 12 of *WCD934x Qualcomm Codec Evaluation Tool* (80-N5283-8 A) for information on the WCD Evaluation Board Engineering View.

6.2 WSA Evaluation Board – Evaluation View

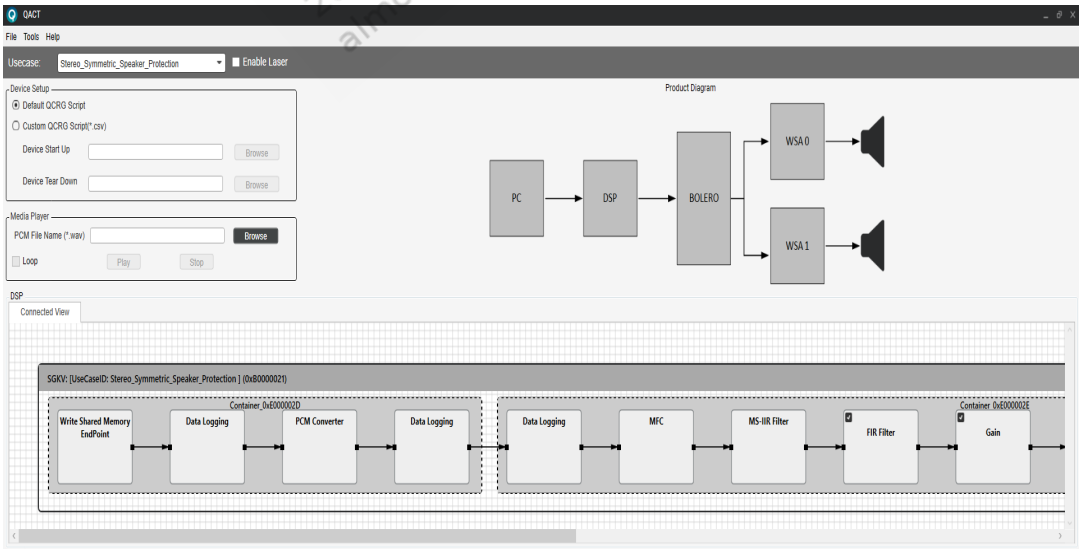
QACT supports playback of PCM/WAV files on speakers connected to the WSA Evaluation Board. QACT automatically detects the WSA Evaluation Board when it is connected to the PC via USB provided that the QTI Thesycon USB drivers are already installed on the PC.

To connect to WSA Evaluation Board:

1. Launch QACT.
2. Connect the WSA Evaluation Board to the PC via USB cable.
3. Click **Connect To Device**.



The following UI should be displayed if the connection to the EVB is successful and the WSA is detected.



NOTE: If multiple devices are connected to the PC, QACT opens a window that lists all detected devices. Select the appropriate EVB hardware from this list and then click **Connect**. The main evaluation view will then be displayed.

This tool can be used to:

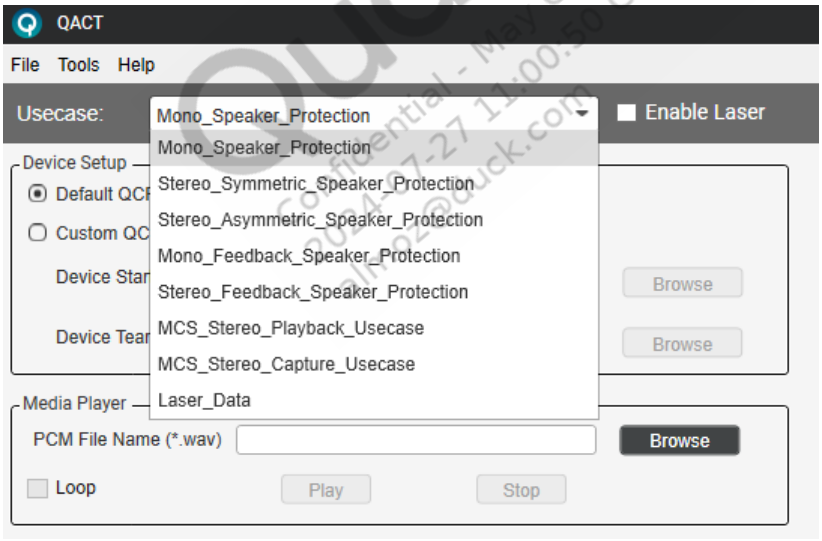
- Evaluate the quality of playback on the speakers attached to the EVB
- Perform RTC of certain modules in the predefined use case(s) with a limited set of QTI postprocessing modules
- Tune and evaluate the performance of the speaker protection algorithm
- Save the tuned ACDB files

NOTE: The tool ONLY supports playback of 16/24 bit mono/stereo PCM/WAV files with a sample rate of 48000 Hz. All other content must be transcoded to this format.

6.2.1 Playback of WAV files using the tool

To start a playback session:

1. Select the appropriate use case from the dropdown based on the number of channels in the file to be played.
2. Click **Browse** and select the WAV file.
3. Click **Play** to start playback.

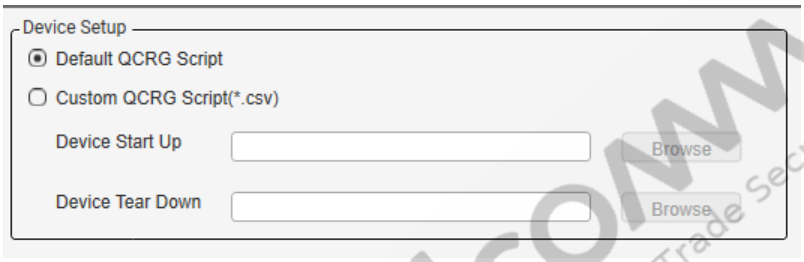


6.2.2 Playback with default/custom QCRG script

There are two options to enable the WSA EVB hardware to set the appropriate hardware registers that enable the speakers/WSA:

- Use the default QCRG script that is shipped along with the tool
- Use a custom QCRG script by specifying the location of the script to be used

Select the appropriate radio button in the main evaluation view based on the requirement. If a custom QCRG script is to be used, provide the location of the script by clicking **Browse** so that it can be executed to enable the hardware before playback is started.



NOTE: Validation of custom QCRG scripts is beyond the scope of QACT. Verify that the script enables the required registers by other means before it is run in QACT. The tool simply runs the script provided and, if playback does not work as expected, checks whether or not the provided QCRG script is valid.

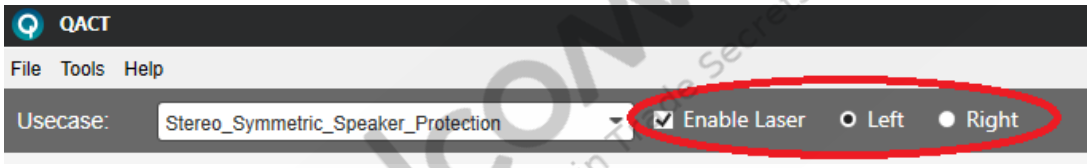
6.2.3 Playback with laser support

If the WSA Evaluation Board has laser measurement capability to measure the excursion of the speaker with greater precision and accuracy, QACT can be used to show the RTM plots of measurement vs. predicted vs. DC excursion.

NOTE: This requires additional hardware that includes a laser measurement setup. The laser should point towards one of the speakers attached to the EVB and the setup should be such that the measured excursion value without any playback is as close to 0 µm as possible.

To use this functionality:

1. Click **Enable Laser** in the main evaluation view.
2. Click **Left** or **Right** based on the speaker to which the laser setup is hooked.



3. Start a playback session using a supported WAV file.
4. Open the calibration view of the speaker protection module in the Connected View tab from the main evaluation view.
5. Navigate to the Monitor section of the selected channel in the Power Amp/Speaker Model block.
6. Click **Start** to see the RTM plots that show the excursion of the speaker as predicted by the algorithm vs. that measured by the laser.

6.2.4 WSA Evaluation Board – Engineering View

To access the registers and see the Engineering View of the WSA Evaluation Board, click **Tools -> Advanced** in the menu bar of the main evaluation view. This opens a new window that shows an engineering view like that described in Section 6.1.

7 QACT latency analyzer

The latency analyzer tool measures the path latency between two (or more) log points in an audio topology. It makes use of the PCM RTM packets generated by data logging modules (that is, data log points). Log points may be placed at different points in a topology to monitor the signal passing through.

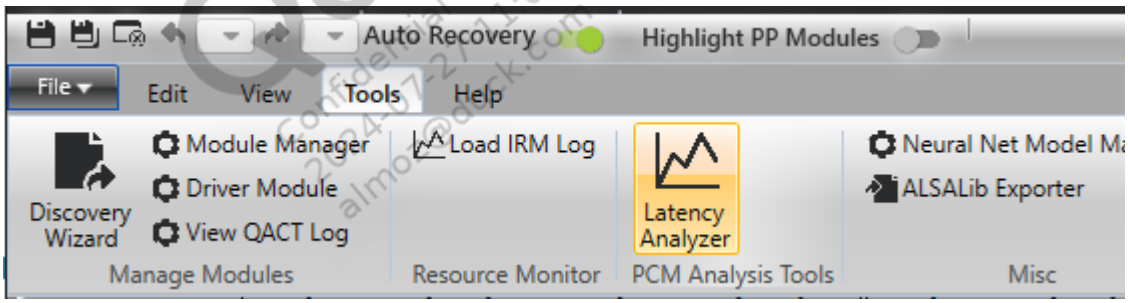
The cross-correlation is used to determine the latency between two PCM signals. It outputs the peak correlation level in percentage (%) and the latency in both samples and milliseconds. PCM data will be captured for each enabled log point and passed to the cross-correlation (with one of the log points being the reference).

The tool will resample a signal if the sample rates between the signals do not match. The lowest sample rate is chosen for resampling.

The tool will also change the bit resolution if the bit width of the two signals do not match. The highest bit width is chosen for the conversion.

7.1 Opening latency analyzer

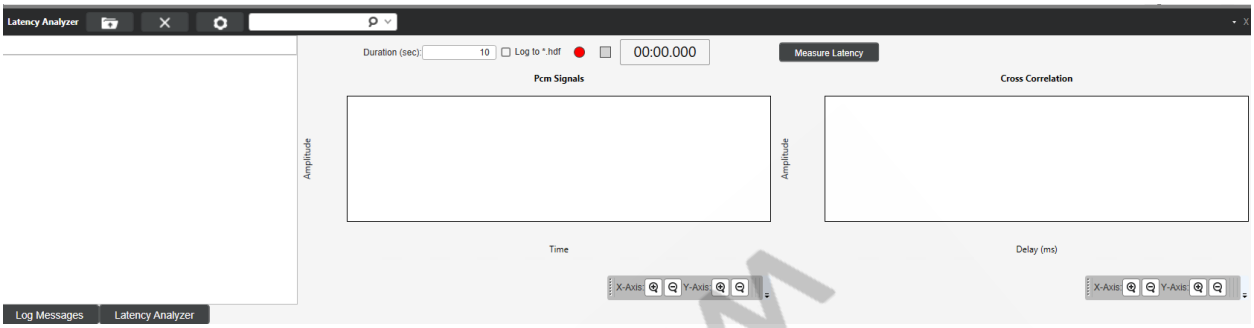
Click **Tools > Latency Analyzer**.



Depending on the mode, two different UIs may be presented: Connected RTC and HDF.

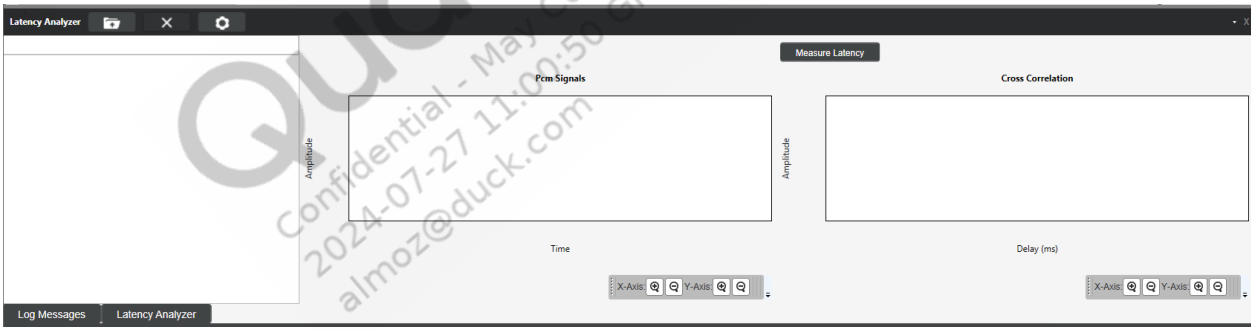
7.1.1 Connected RTC

This mode enables real-time capture of PCM data from an active use case (playback, record, VoIP, etc.) using the media controls above plots. The user can generate HDF files for offline viewing using the “Log to *.HDF” control.



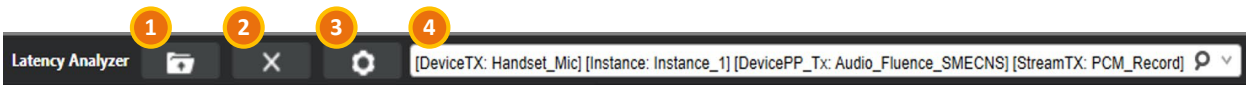
7.1.2 HDF

A HDF file contains the captured PCM RTM packets for the enabled data logging modules. This mode allows the user to open an HDF file which will be parsed and display the log point PCM data captured during the RTC session. There are no media controls in this mode.



7.2 UI controls

7.2.1 Title bar controls



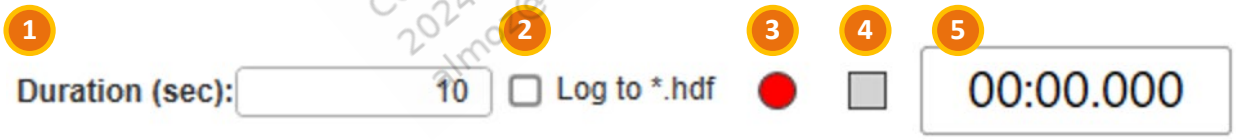
#	UI Control	Description
1	Open HDF File	Opens a file dialog where the user can select a *.hdf/*.isf file from the windows explorer. The HDF file will be parsed, and the signal list will be populated with data logging modules whose log codes were enabled in the use case.
2	Close HDF Session	This control is only enabled after opening an HDF file. It closes the current HDF session by clearing the signal list and closing the HDF file.
3	Settings	Opens the options dialog directly to the latency analyzer page. Additional settings for the tool can be modified here.

#	UI Control	Description
4	Active use case dropdown	Used to filter the signal list by showing all data logging modules associated with the selected graph key vector

7.2.2 Options

Allows the user to set the latency analyzer output directory where all output of the Latency Analyzer will be written.

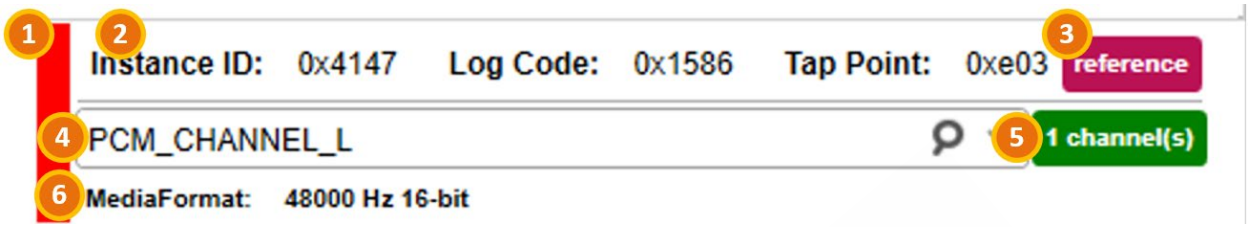
7.2.3 PCM capture controls



#	UI Control	Description
1	Duration	The length of the capture in seconds. The maximum duration allowed is 1 min. The minimum duration is 10 secs.
2	Log to HDF	Checking this option will generate an HDF file in the latency analyzer output directory.
3	Record	Starts audio capture and runs for the duration specified.
4	Stop	Stops the current capture session.
5	Time display	The timer displays the current time in mm:ss.sss format (minutes and seconds)

7.2.4 Data logging signal list

This is a list of data logging modules filtered by the active use case dropdown or parsed from the HDF file.

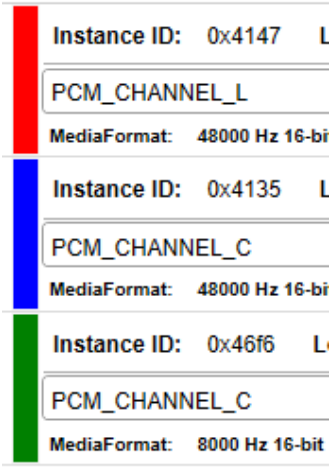


#	UI Control	Description
1	Plot color	The color of the signal used in both the PCM signal and cross-correlation plots
2	Instance ID, Log Code, Tap Point	The module instance ID that identifies the module instance along with the data logging configuration (log code + tap point). The module instance ID is not available in HDF mode due to SPF framework limitations.
3	Reference Label	A label indicating that the signal should be used as the reference point when calculating the latency. To set a reference signal, right click on a data logging module in the list. This reveals a context menu with an item in the menu “Set as Reference Signal”.
4	Channel Selection Dropdown	A dropdown containing a list of available channels
5	Channel Count	Indicates the number of channels captured for the data logging module
6	Media Format	The sample rate and bit width of the PCM signal passing through the data logging module

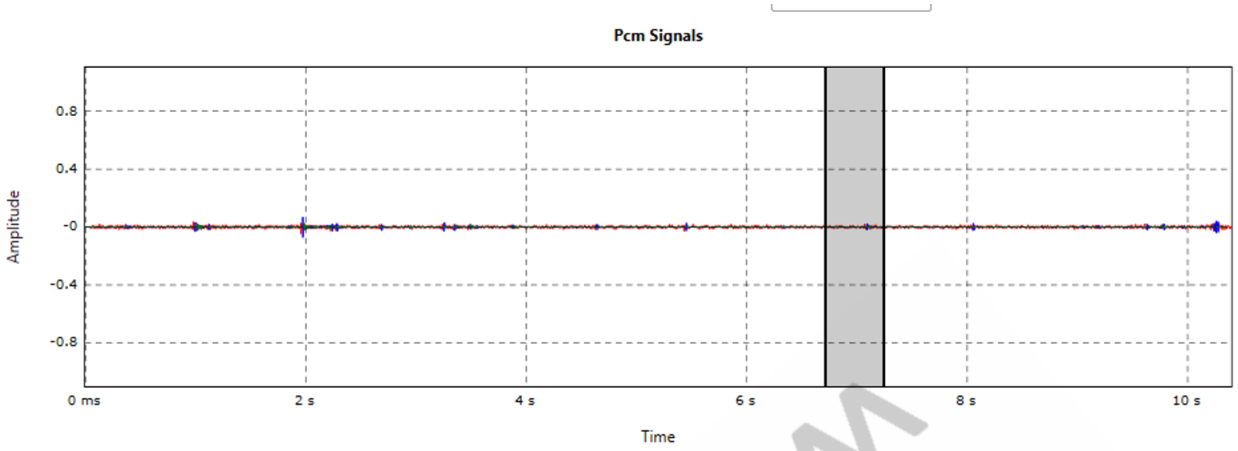
7.2.5 PCM and cross-correlation plots

The PCM captured during the active use case or parsed from the HDF file are displayed on the PCM plot. The plot colors shown in the data logging signal list are used to associate the plotted signal with the log point in the list.

PCM can be selected and then analyzed using the Measure Latency button, which will calculate the latency between the log points using the cross-correlation.



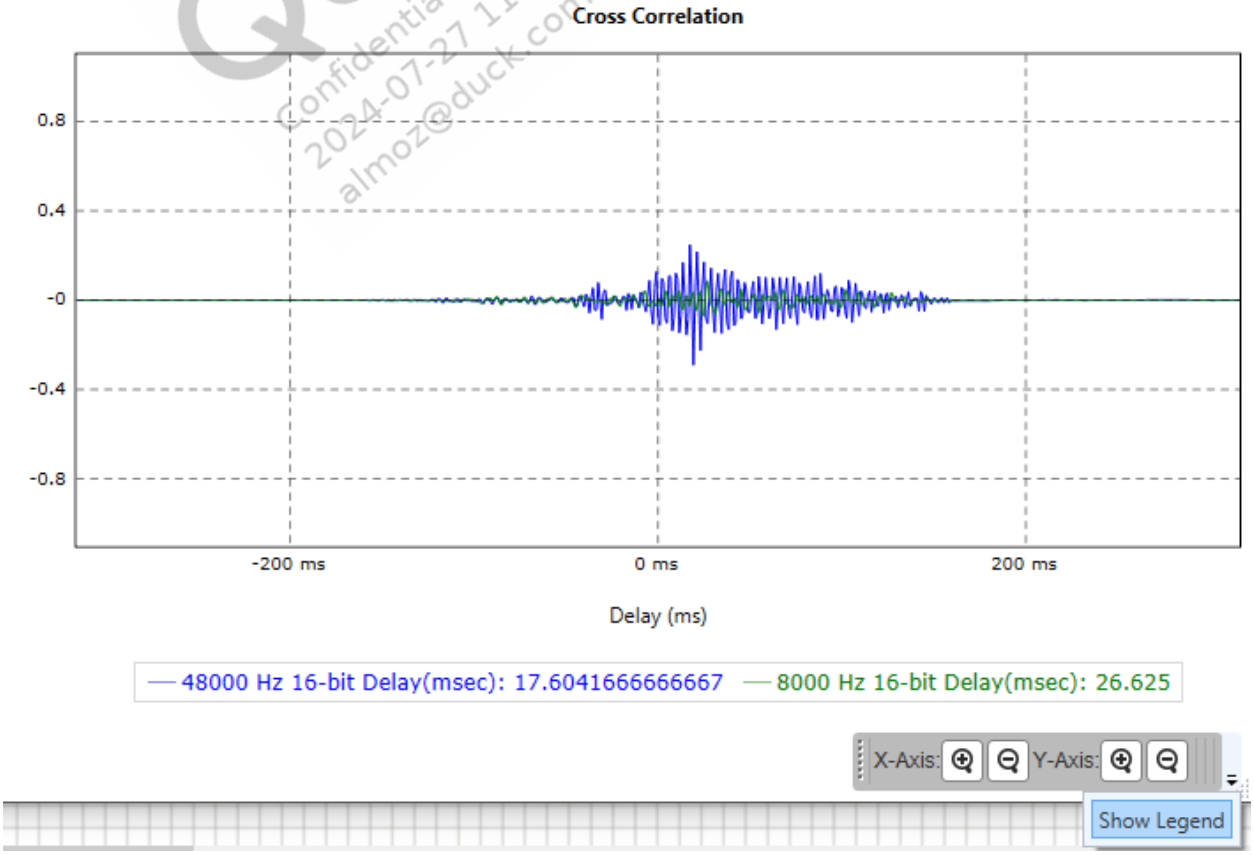
PCM signal plot



A single plot where all the PCM signals are plotted and overlayed. Users can select a range of PCM to analyze by dragging and dropping a selection rectangle.

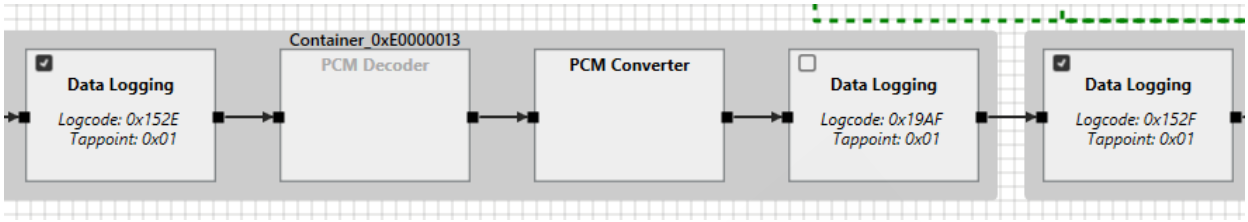
Cross-correlation plot

The cross-correlation of two PCM signals are plotted here. The plot color of the non-reference signal(s) are used. To view the calculated latency, click the dropdown on the toolbar and select **Show Legend** on the context menu. The legend will display below the plot.



7.3 Workflow

Use the RTC or HDF workflow depending on your needs and then follow the measuring latency workflow described in Section 7.3.3



7.3.1 RTC workflow

1. Open QACT and connect to the device.
2. Click **Tools > Latency Analyzer.**
3. Start a use case on the device.
4. Once QACT displays the use case in graph view, select a use case in the latency analyzer use case dropdown.
5. The signal list should be populated with data logging modules that are enabled in the use case.

   Data logging modules can be enabled/disabled in graph view. You will need to refresh the signal list by clearing the selected signal using backspace and reselect the use case.
6. Configure two or more data logging modules by selecting a channel to display.
7. Choose one of the data logging modules to be the reference signal by right clicking an item in the list and selecting **Set as reference** in the context menu.
8. Specify the duration of the capture.
9. Optionally check the “Log to *.hdf” checkbox to create an HDF file for offline viewing.
10. Start the capture session by clicking **Record**.
11. After the capture session is complete, the PCM for all data logging modules will be displayed in the PCM Signal Plot.

7.3.2 HDF workflow

1. Open QACT in Offline/Connected mode.
2. Click **Tools > Latency Analyzer**.
3. Click **Open HDF**.
4. Select an HDF file from your file system. The HDF file is parsed, and the signal list should be populated with data logging modules that were enabled in the use case.
5. Configure two or more data logging modules by selecting a channel to display.

The PCM for the selected channel will be displayed in the PCM Signal Plot

7.3.3 Measuring latency

1. Select a range of PCM in the PCM plot. Ideally, this should be less than 2 secs worth of PCM. Any longer and the processing time of the cross-correlation will go up.
2. Click **Measure Latency**. The cross-correlation plot will be updated.
3. To view the calculated latency, click the dropdown on the toolbar underneath the cross-correlation plot and select **Show Legend** on the context menu. The legend will display below the plot.

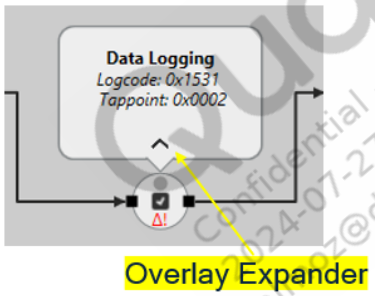
8 Data logging PCM viewer

The PCM data that the data logging module receives in RTC mode is displayed in a waveform with the help of the PCM viewer. The waveform can be seen in three ways:

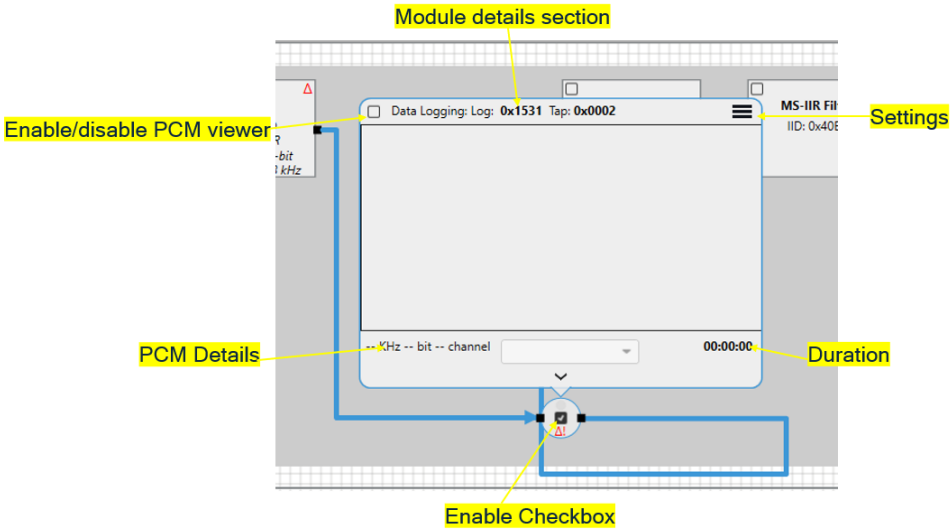
- Data logging module overlay (RTC)
- Real-time PCM viewer in spec view
- Static mode

8.1 Data logging module overlay

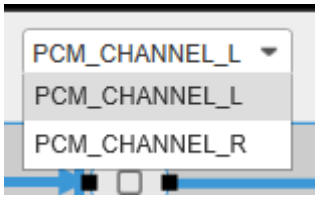
- In RTC mode graph view, the data logging module has the functionality of displaying or collapsing the overlay. If collapsed, it would be the same size as that of other modules, otherwise it expands as a larger control which is used to view waveform.
  - Collapsed state:



  - Expanded state:

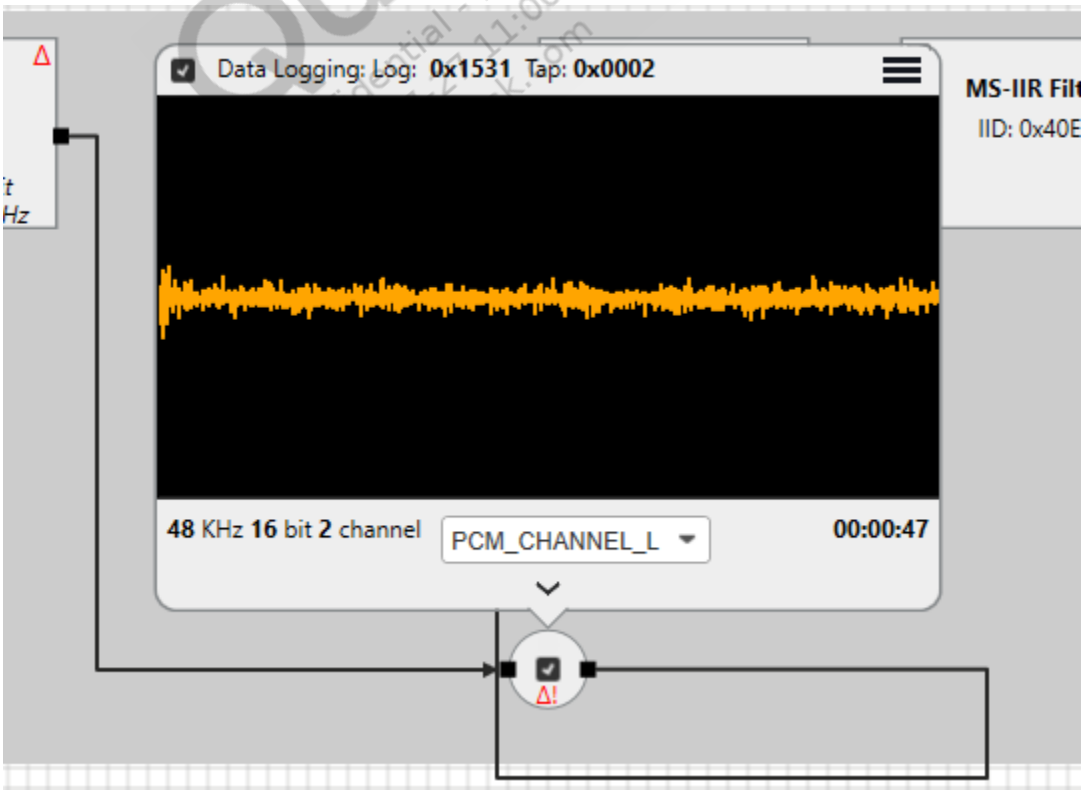


- The following are displayed on the data logging overlay:
  - Module details – Consists of name, log code, and tap point of the data logging module
  - Enable/disable PCM viewer checkbox – Shows or hides the waveform. When the checkbox is checked, PCM packets are subscribed on log code and tap point. Unsubscribed when unchecked.
  - PCM details section – Displays sample rate, bit width, number of channels and channel selection dropdown



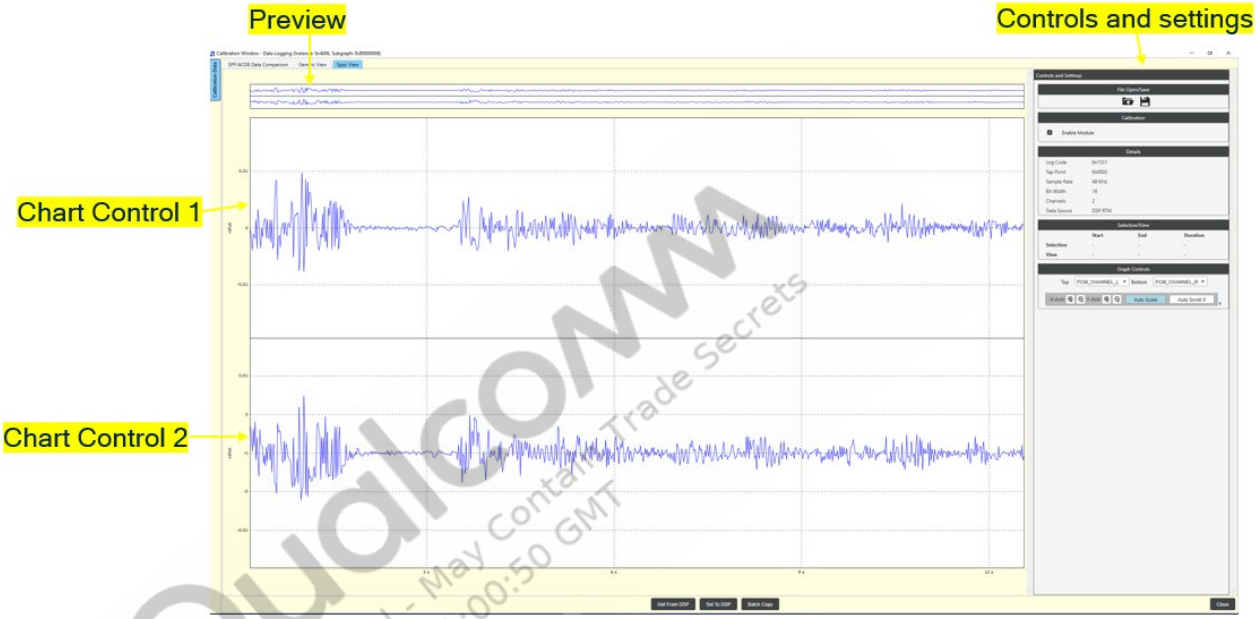
  - Duration section – Displays the time duration in which the module's PCM viewer is enabled
  - Settings – Contains options to set the display ratio (number of packets per pixels; time taken for waveform to each from start to end of overlay display)
  - Enable checkbox – For calibration. The functionality is the same as that of the check box present in the top left corner of other modules.

Example:



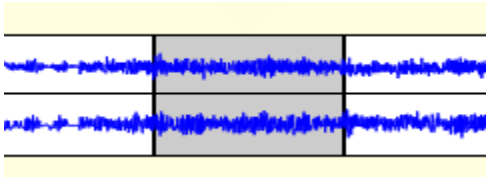
8.2 Real-time PCM viewer in spec view

In RTC mode, to view the waveform in spec view, first the PCM viewer should be enabled in the overlay and the spec view should be open. This hides the waveform in the overlay. At most two chart controls will be shown in spec view, each chart control to show PCM waveform for each channel.

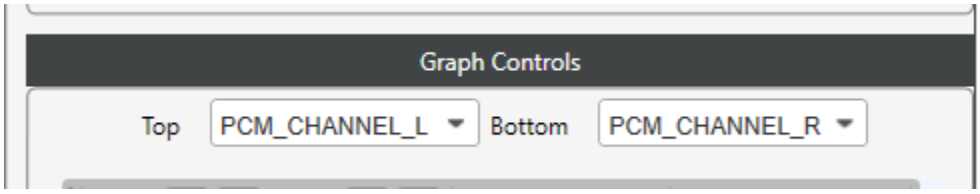


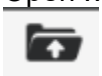
Preview

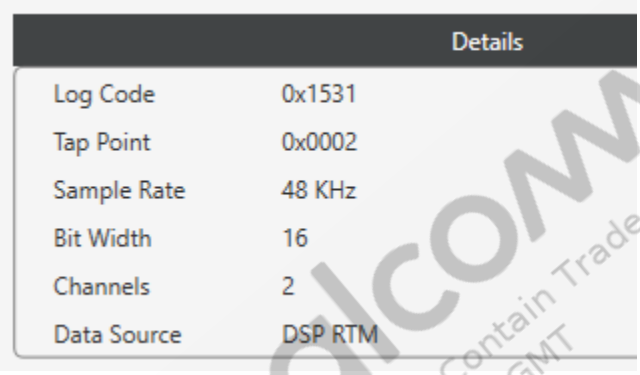
- The graph of two channels that are selected in the channel dropdown in controls and settings
- Selecting a region in preview can be done by clicking at a point and dragging to the required position



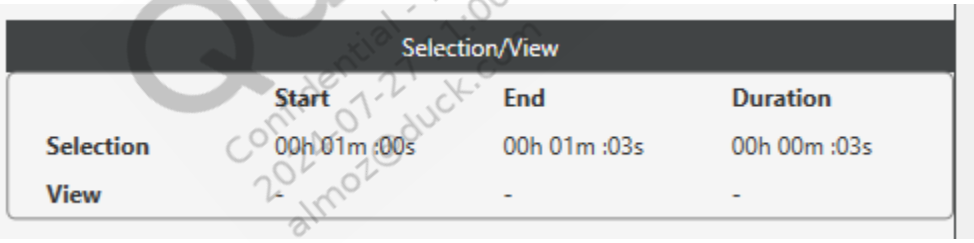
- Chart control 1 – The PCM data of the “Top” channel selected in controls and settings
- Chart control 2 – The PCM data of the “Bottom” channel selected in controls and settings



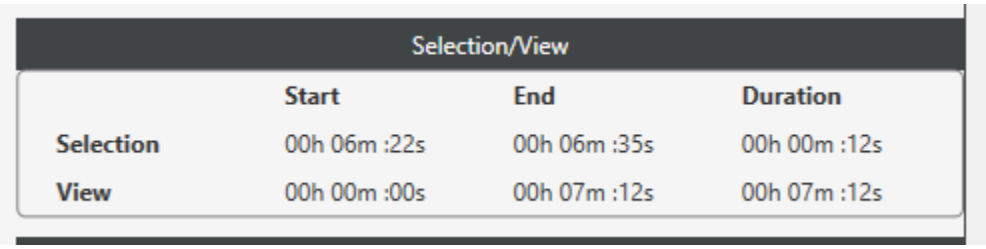
- Controls and settings:
  - Open .wav file – Open any wave file in the data logging spec view by clicking 
  - Save File – In RTC mode, when the PCM viewer is running on real-time PCM packets, the packets that are logged so far in the current session can be saved as a .wav file. Opening the spec view starts a session and closing ends the session.
  - Details – Displays the details of PCM data including sample rate, bit width, number of channels and log code, and tap point of the data logging module



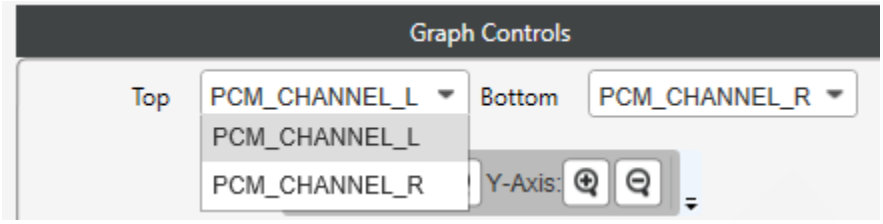
- Selection – Displays the selection positions that are selected in the preview



- View – Shows the duration of the input .wav file



- Channel selection – The incoming PCM data can have multiple channels. But only the channels that are displayed in the dropdown are visible in the corresponding chart control



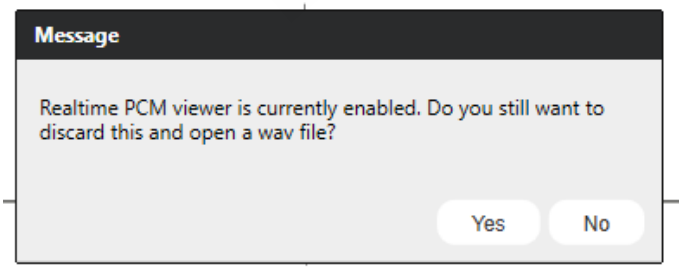
- Zoom controls – Zoom in and zoom out for X and Y axis.

- Auto Scale – The graph scales in the chart controls (that is, at any point all the samples are visible on UI)
- Auto Scroll – The graph scrolls with regard to the X axis. Only a set of samples are displayed at any point.

8.3 Static mode

A .wav file can be imported in data logging spec view to view the contents of the file. This can be done in offline and online modes of QACT. This is called static mode because once a file is opened, the data does not change.

- In RTC mode, if real-time PCM viewer is running, the user can still open a .wav file but the real-time PCM data is gone. The user needs to confirm this step.



- Save option is hidden in static mode
- Graph controls like auto scale and auto scroll are hidden in static mode
- The opened wave file name can be seen in data source of details section

Part 4: Support and References

A Requesting support

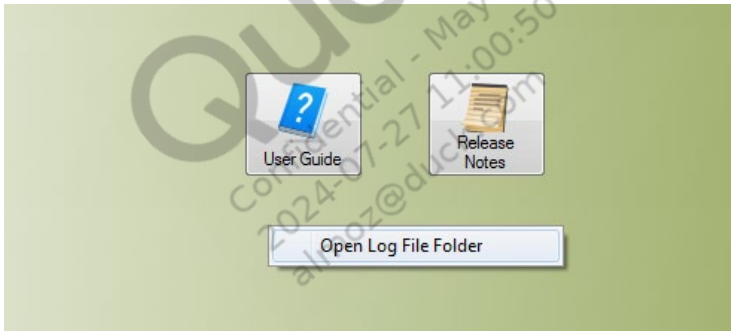
A.1 QACT log

QACT creates log files to store log messages when a user is tuning data. These log files allow QACT developers to perform analysis when a QACT-related issue is reported.

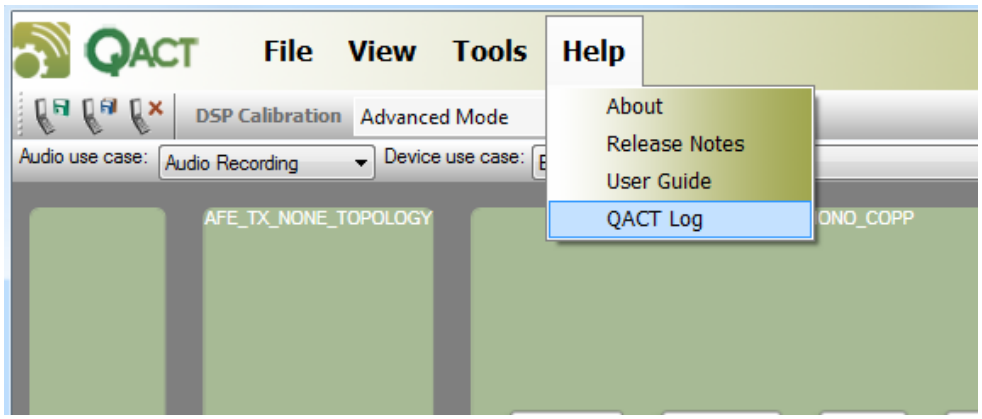
After QACT starts, once a file is opened or a device is connected, a log file for this QACT session will be created with the name format QACT_version_timestamp.log. This log file will be used to store information until the QACT program is exited (not when a file is closed).

All QACT log files are stored in \QACT installation folder\QACTLogs. You can access the log folder in QACT in two ways:

- Right-clicking anywhere in the QACT window and selecting **Open Log File Folder** while in the QACT main screen.



- Select **Help>QACT Log** after opening a file or connecting a target.



A.2 Use space logs

NOTE: This section applies to LA only.

To get the user space logs, run the following command:

```
adb shell logcat > logcat_log.txt
```

If you have QACT connection issues, see messages about ACDB not being initialized, or data from .acdb files does not seem to be set, you will also want to kill the mediaserver during user space logging.

To restart mediaserver, run the following commands:

```
adb shell
ps "mediaserver"
kill -9 <pid>
```

A.3 Kernel logs

NOTE: This section applies to LA only.

To get the kernel logs, run the following commands:

```
adb root
adb shell
mount -t debugfs debugfs /sys/kernel/debug
echo file audio_calibration.c +p > /sys/kernel/debug/dynamic_debug/control
echo file audio_cal_utils.c +p > /sys/kernel/debug/dynamic_debug/control
echo file rtac.c +p > /sys/kernel/debug/dynamic_debug/control
echo file q6adm.c +p > /sys/kernel/debug/dynamic_debug/control
echo file q6afe.c +p > /sys/kernel/debug/dynamic_debug/control
echo file q6voice.c +p > /sys/kernel/debug/dynamic_debug/control
echo file q6asm.c +p > /sys/kernel/debug/dynamic_debug/control
exit
adb shell cat /proc/kmsg > kernel_log.txt
```

A.4 QXDM logs

To get the QXDM logs:

1. Open QXDM.
2. Click **Options > Message View Config**.
3. Under **Known Messages**, select the following checkboxes:
   - QDSP6
   - APR modem
   - APR apps
   - APR ADSP
   - Audio vocoder services
   - Voice recognition
4. Click **Options > Connections**.
5. Set the Target Port field to the connected device and click **OK**.
6. Start the use case you are currently debugging.
7. When done capturing logs, click **Options > Connections**.
8. Select Disconnect and click **OK**.
9. Click **File > Save** Items to save the .isf file. Provide this file for debugging when submitting a case.

B Debugging

B.1 QACT connection

This section explains issues that may be encountered when using QACT.

QACT installation corrupted. Please reinstall QACT.

Critical DLLs needed for QACT functionality are not found in the installation directory. To resolve this issue, uninstall QACT and reinstall to ensure all files are in the installation directory.

Unable to find any connected device

- Possible issue 1 – QPST is too old and needs to be upgraded
  - If this is the issue, QPST will look like it has detected the device, but the device name has question marks in the name.
  - To fix this issue, upgrade QPST and try again.
- Possible issue 2 – QACT and QPST are not run in the same mode (Administrator vs. User mode
  - If this is the issue, ensure that both QACT and QPST are run in Administrator mode.
- Possible issue 3 – If device is not detected in QPST, confirm the following:
  - Device is connected to PC
  - USB driver is installed/working

Unable to receive any data from the device with %s on port %s

- Possible issue 1 – QACT's communication service does not have permission to access DIAG. To fix, run adb commands to give access.
- Possible issue 2 – QACT's communication service failed to initialize. Provide QACT and userspace logs and refer to Appendix A to request QTI debug support.

Unable to receive files from the device

QACT is unable to find the QWSP and ACDB files on the device. Provide QACT and userspace logs and refer to Appendix A to request QTI debug support.

Unable to open the downloaded files from the device

QACT is unable to find the QWSP and ACDB files on the device. Provide QACT and userspace logs from the device and refer to Appendix A to request QTI debug support.

Unable to complete file download from the device

QACT encountered errors during the file download procedure. Provide QACT and userspace logs from the device and refer to Appendix A to request QTI debug support.

B.2 Discovery Wizard

This section explains issues that may be encountered when using QACT's Discovery Wizard.

B.2.1 Error in parsing file(s)

One or more file(s) are not in supported format. Cannot import them.

Open the QACT log and refer to Appendix A. In the log file, search for the following messages for details. For further information, request QTI debug support and provide the QACT log.

Log message	Meaning
Invalid XML file format for file	Imported file is not in valid XML format. Open it in an XML editor to fix it.
Invalid definition XML file format for file	Imported XML file does not contain any h2xml definition

There is some invalid data

Open the QACT log and refer to Appendix A. In the log file, search for the following messages for details. For other error messages, request QTI debug support and provide the QACT log.

Log message	Meaning
Invalid XML file: PLATFORMS node not present!	PLATFORMS is required for a key definition. It is missing from the imported key definition file.
Duplicate key	Some key definition is duplicated in the imported key definition file
PROCESSORS information is missing in module definition	Each module definition must have the supported processor(s) provided. It is missing. Check if the "-a @h2xml_processors" argument is missing in the H2XML command while converting header file to xml file.
Invalid processor type for a module definition!	A module's supported processor type is invalid
Could not find module ID	Module ID is missing in module definition
Could not find module name	Module name is missing in module definition
Could not parse module info	Module meta info contains invalid data in module definition
Could not parse params. Ignoring parsing for this module!	Param definitions in a module has some invalid information

Log message	Meaning
Could not find param ID	Parameter ID is missing in a parameter definition
Could not find param name	Parameter name is missing in a parameter definition
Could not find max size	maxSize is missing from a parameter definition. maxSize is required for a PID definition. It should be the payload size of a PID.

The following files have modules which are not licensed for tuning

Some imported module(s) IDs are not in a range of licensed IDs. Provide QACT logs and refer to Appendix A to request QTI debug support.

B.2.2 Failures exist (red text) in the summary page's change summary

Error in importing definitions

There is some error while importing definition(s) to ACDB. Provide QACT logs and refer to Appendix A to request QTI debug support.

Unable to update data due to the following error

There is some error while importing definition(s) to ACDB. Provide QACT logs and refer to Appendix A to request QTI debug support.

C Elite-to-AudioReach data migration

There are many modules which are common in both Elite and AudioReach. Tuning efforts are less significant if calibration data for these common modules can be migrated from Elite to AudioReach. The Data Migration Wizard can be used to migrate this calibration data.

C.1 Use case map

The ACDB file format of Elite and AudioReach are different. To migrate calibration data successfully, QACT needs mapping information. This information can be found in the use case map .xml file, which contains three important nodes:

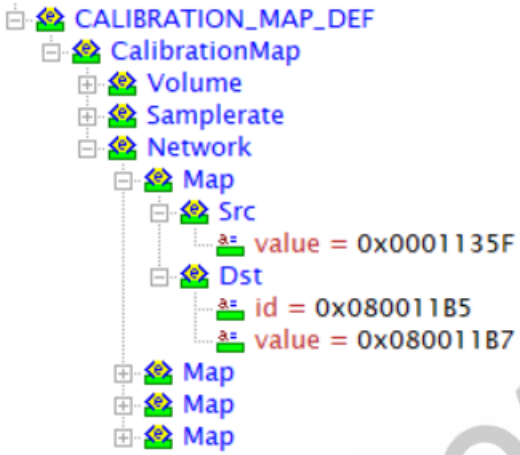
- Use case map def – This node has the information that maps the device/device pair in Elite to the use case in AudioReach.

  The Src node specifies the device/device pair in the Elite ACDB files. The Dst node specifies the GKV in the AudioReach ACDB files. The following is an example.

```
<UsecaseMap shouldExactlyMatch="false">
 <!-- Audio playback use case (Mono) -->
 <Src>
  <!-- A set of Elite usecase keys -->
  <Key name="Device" value="0x00000027" />
  <!-- DeviceRX: BT_SCO_WB_SPKR -->
  <Key name="AppTypeCopp" value="0x0001113A" />
  <!-- AppType: Voip_Audio -->
 </Src>
 <Dst>
  <!-- Graph Key Vector from reference ACDB File -->
  <Key name="0xA2000000" value="0xA2000003" />
  <!-- DeviceRX: BT_Rx -->
  <Key name="0xB4000000" value="0xB4000001" />
  <!-- BtProfile: SCO -->
  <Key name="0xB5000000" value="0xB5000004" />
  <!-- BtFormat: SWB -->
 </Dst>
 <Modules>
 </Modules>
</UsecaseMap>
```

- Calibration map def – This node helps QACT map the CKVs in AudioReach to indices in Elite.

  The Src node specifies the index value in the Elite ACDB files. The Dst node specifies the CKV ID and value in the AudioReach. The following is an example.



- Driver map def – This node helps QACT to migrate data for driver-specific modules.
  - The Src node contains the table ID and the indices of that table ID from which data has to be migrated
  - The Dst node contains the key value for the driver module
  - The modules node specifies the list of modules that need data migration.

  The following is an example driver map node.

```
<DRIVER_MAP_DEF>
    <DriverMap shouldExactlyMatch="false">
    <!--<Src>
            <!{1}** A set of Elite Table Id **{1}>

            <Key name="module_id" value="0x00011278"/>
        </Src>-->

        <Src>
            <!-- A set of Elite Table Id -->
            <Key name="Table_id" value="0x00000003"/>
            <Key name="dev_id" value="0x00000059"/>
            <Key name="appType_id" value="0x00011130"/>
            <Key name="Volume" value="0x00000000"/>

        </Src>
        <Dst>
            <!-- Graph Key Vector from reference ACDB File -->
            <Key name="0xAF000000" value="0xAF000001"/>
            <!-- DevicePP_Rx: Audio_LL_Default_PP -->
        </Dst>
        <Modules>
            <Module srcName="AUDPROC_MODULE_ID_VOL_CTRL" srcMid="0x00010BFE" dstName="AUDPROC_MODULE_ID_VOL_CTRL" dstMid="0x00001234">
            </Module>
        </Modules>
    </DriverMap>
```

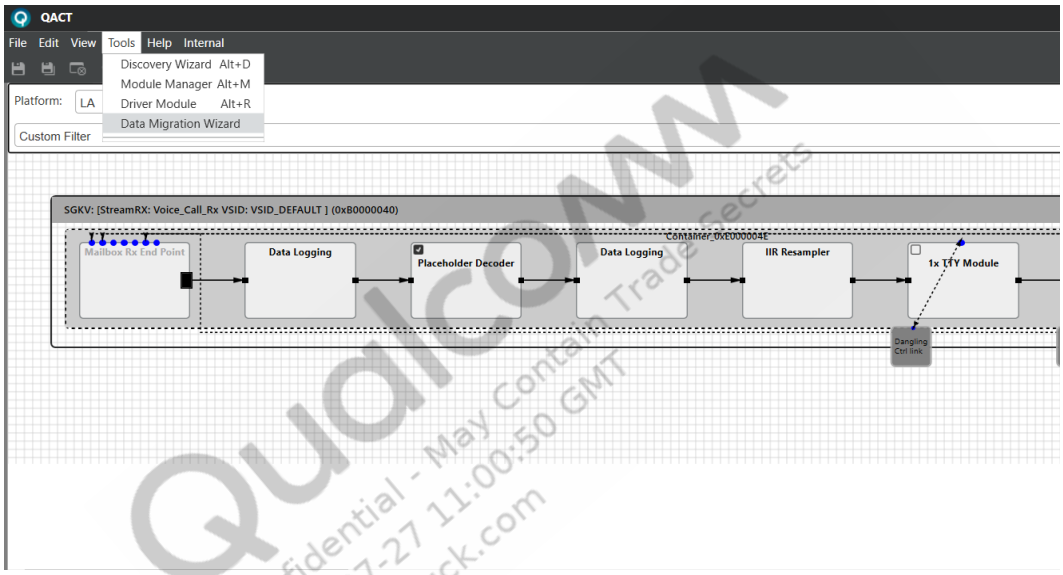
C.2 Data migration

Data migration can be performed using either of the following two methods.

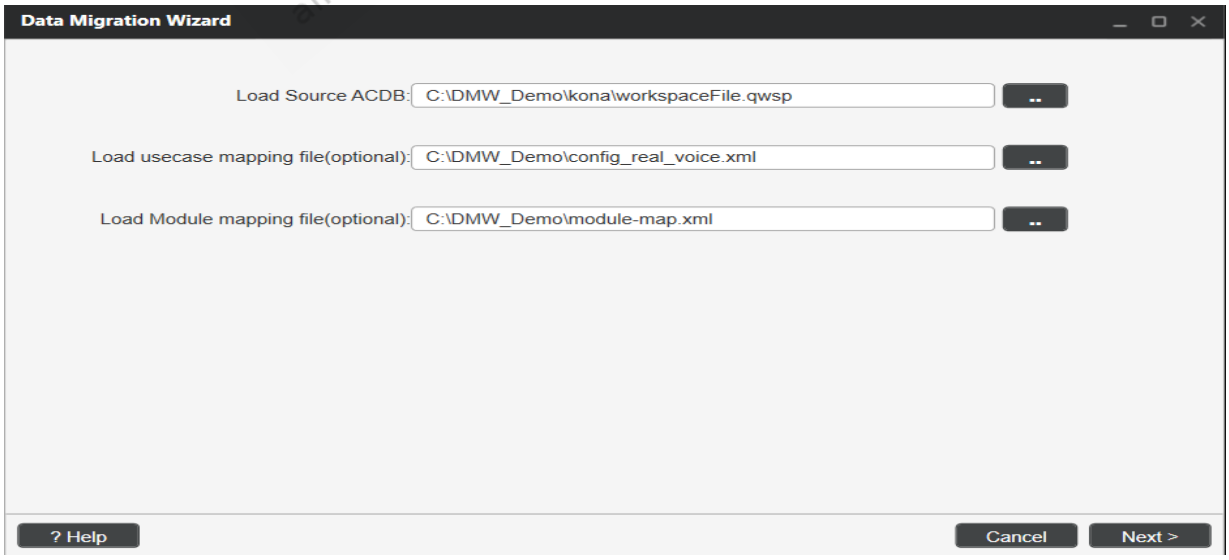
C.2.1 User Interface

To migrate data via the user interface:

1. Open AudioReach ACDB files.
2. Select **Tools -> Data Migration Wizard**.



3. Use the respective **...** buttons to select the input Elite ACDB files (Load Source ACDB), use case mapping file, and module mapping file.



4. Click **Next**.

C.2.2 Command line interface

A console application, QACTUtility.exe, is provided with the QACT installation and can be found in the installation folder.

```
QACTUtility.exe data_migration <Elite_QWSP> <AudioRach_QWSP> [-umap
usecase_map.xml] [-mmap module_map.xml]
```

D QACT v8.0 to v8.1 file conversion

QACT v8.1 requires more switch-related meta information in the workspace file. While opening QACT v8.0 compatible workspace files, a conversion procedure is triggered to automatically insert switches into graphs by a predefined policy. This conversion will make opening a QACT v8.0 workspace file take more time than opening a QACT v8.1 compatible workspace file. This conversion only converts workspace files, not *.acdb files. Once a workspace file is converted, it cannot be opened in QACT v8.0 anymore.

NOTE: All inserted switches are non-concurrent (Input Selector and Output Selector) since, in QACT v8.0, there is no concurrency information stored in workspace file. QACT v8.1 cannot determine if a use case should be concurrent or not. Adding non-concurrent switches by default will not have any limitation in future graph design. If system designer needs to set a switch to concurrent, change it in property view (refer to Section 4.2.2.4).

D.1 Policy to insert switches into graphs

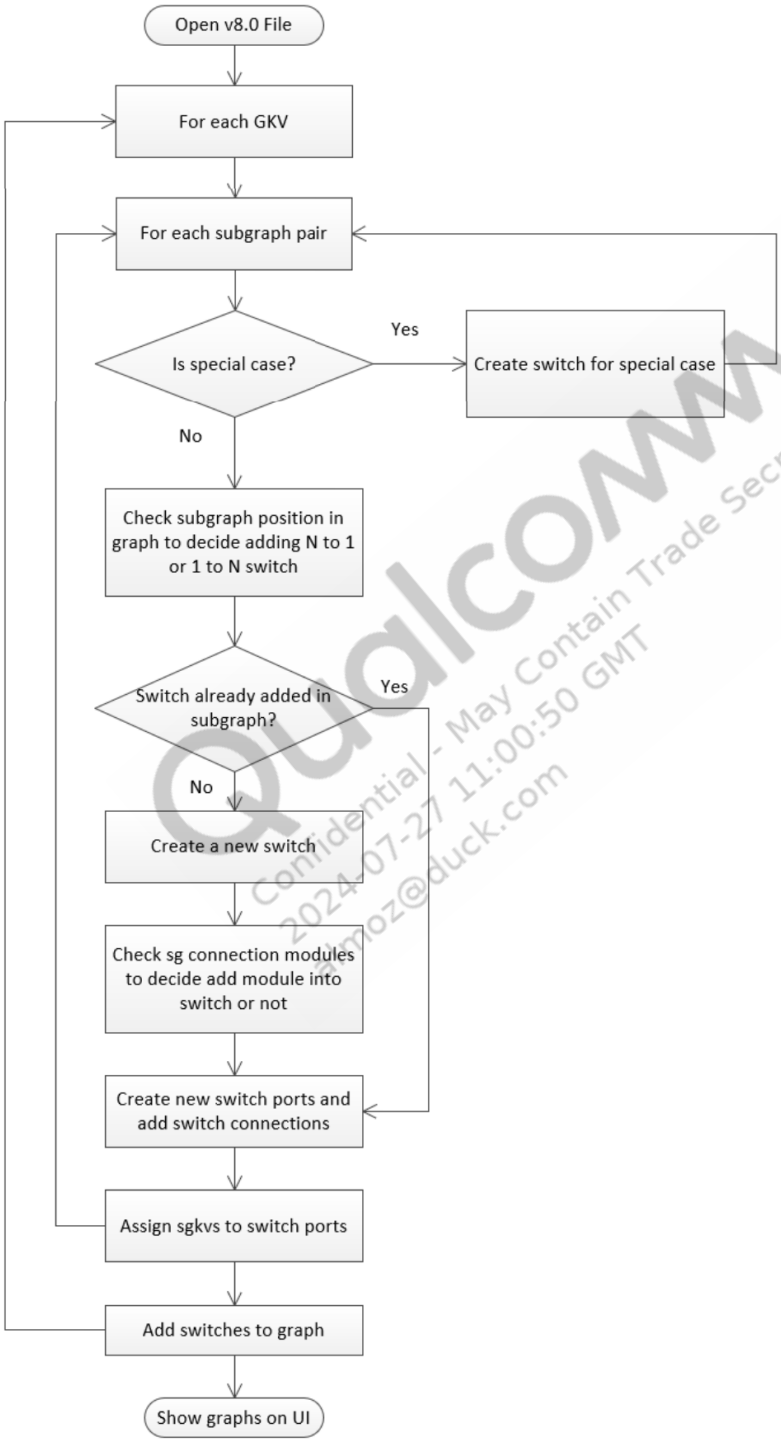
QACT v8.1 has an internal policy to insert switches automatically while opening an old QACT v8.0 workspace file. The policy includes:

- Convert graphs one-by-one based on the order of GKVs in the ACDB file.
- For each graph, try to insert switch(es) based on subgraph pair one-by-one.
  - Source and destination subgraph positions in the graph are considered (Source EP – middle, middle – sink EP, middle – middle, source EP – sink EP).
  - In source and destination subgraphs, the subgraph connection's source module and destination module are considered (Splitter, Splitter Render, Sync module, Sal, Accumulator&Limitter, etc.). These special modules will be added in a switch. Otherwise, the switch will not contain any module.
  - Source and destination subgraph's SGKVs are used to assign KVs to switch ports
  - Some subgraph pairs are considered as special cases, such as EC, Voice Activation, etc. These subgraph pairs will be detected and have switches created in special ways.
- For a subgraph shared by multiple graphs, need to consider impact of other graphs while adding/updating a switch in this subgraph.
  - If a subgraph is shared by both graph 1 and graph 2, while converting graph 1, there is no need to add a switch in this subgraph per policy. While converting graph 2, a switch is added in this subgraph. QACT will update graph1's meta data as well by adding the switch and update switch connections for graph 1.
  - If a shared subgraph has more than one SGKVs for different graphs, its switch port may have more than one KVs as well.

- If a graph only contains one subgraph, QACT cannot decide what switch should be added. Mark it as a manual GKV graph (refer to Section 4.4).

Since there are multiple ways to add switches in graph, QACT v8.1 inserts switches based on the above policies. The system designer may have different preferences to use switches in a graph. If needed, the system designer should update graphs manually in QACT v8.1 after opening the file.

D.2 Workflow to create switches



D.3 Create switch for special cases

For a given subgraph pair, first check if it is a special cases. QACT predefines some special cases that cannot have switches created using generical workflow. For these special cases, switches will be created specially.

D.3.1 Voice activation special case

In a voice activation scenario, there are two connections between the subgraph pair. One connection is from Audio_Dam_Buffer. The other one is from Splitter. QACT will add a 1 to N switch in source subgraph and add both Audio_Dam_Buffer and Splitter in the switch.

D.3.2 EC special case

All EC use cases starts from a Splitter module in source subgraph of a subgraph pair. The Splitter module also has a connection to another module in same subgraph. As long as such a subgraph pair is detected, an EC switch is added between the two subgraphs of the subgraph pair. Source subgraph's SGKV will be assigned to EC switch's input port, and destination subgraph's SGKV will be assigned to EC switch's output port.

D.3.3 ACD special case

All ACD use cases have subgraph pair that a Splitter module in source subgraph connects to Audio Context Detection (ACD) module in destination subgraph. The user should add a 1 to N switch containing a Splitter module in the source subgraph and a N to 1 switch in the destination subgraph and connect to ACD module. Source SGKV is assigned to source switch's input port as KV. And destination SGKV is assigned to destination switch's output port as KV.

D.3.4 Voice activation FFEC special case

In this special case, the subgraph pair's connection is from the Splitter module to the Audio Dam Buffer module. However, the source subgraph contains two Splitter modules. The other Splitter is used for the Voice Activation special case and a switch is already added based on it. There is no need to add another switch in source subgraph. In this special case, there is no need to add any switch in either source or destination subgraph.

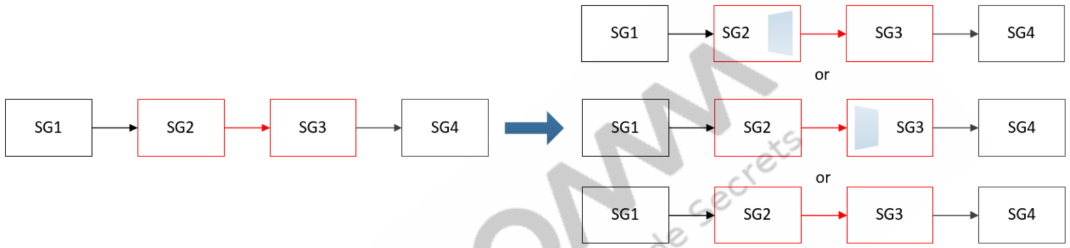
D.4 Create switch for non-special cases

If a subgraph pair is not a special case, QACT will determine a switch based on source and destination subgraph positions in the graph and modules of the subgraph pair connection.

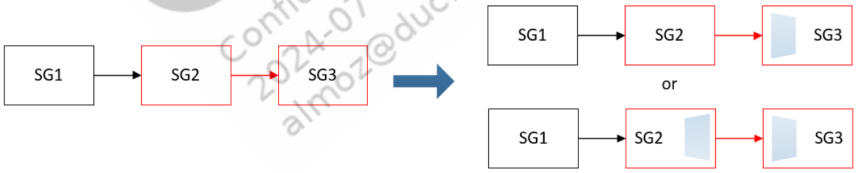
There are four types of scenarios: source EP to middle, middle to middle, middle to sink EP, and source EP to sink EP.

- Source EP to middle: an Output Selector is added in source subgraph.
  - If a source module of a subgraph pair connection is Splitter or Splitter Render, add the module inside the switch. Otherwise, no module should be added in switch.

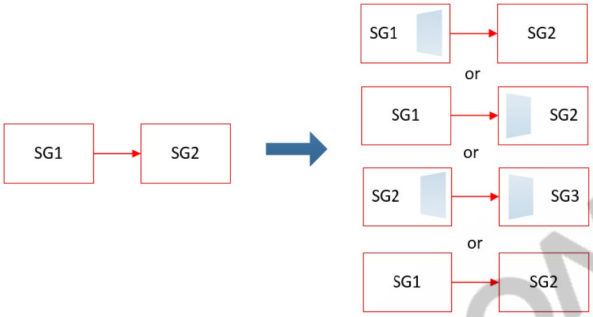
- Middle to middle:
  - If a destination module of a subgraph pair connection is Sync or Accumulator&Limiter, add an Input Selector switch in the destination subgraph and add the module in the switch.
  - If destination module of a subgraph pair connection is not Sync or Accumulator&Limitter, check its SGKV:
    - If SGKV is not for voice, add an Input Selector switch in the destination subgraph.
    - If SGKV is for voice, add an Output Selector switch in the source subgraph.



- Middle to sink EP: an Input Selector is added in the destination subgraph.
  - If the destination module of subgraph pair connection is Sync or Accumulator&Limitter, add the module inside the switch. Otherwise, no module should be added in the switch.
  - If the source module of a subgraph pair connection is Splitter or Splitter Render, and the source subgraph has no switch yet, add an Output Selector switch in the source subgraph, and add Splitter or Splitter Render in the switch.



- Source EP to sink EP: check the source and destination modules of the subgraph pair connection to determine the switch.
  - If the source module is Splitter or Splitter Render, add an Output Selector switch in the source subgraph and add the module in the switch. Otherwise, no switch is added in the source subgraph.
  - If the source module is Sync or Accumulator&Limiter, add an Input Selector switch in the destination subgraph and add the module in the switch. Otherwise, no switch is added in the destination subgraph.



D.5 Create new switch port

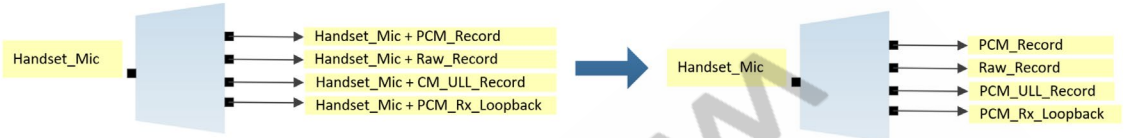
For a shared subgraph between graphs, if it has a switch added per previously parsed GKV, when the second GKV is being handled, the same switch should be updated by creating new ports and connections (both internal and external) based on new subgraph pair connections. Switch ports will be created by order of parsed subgraph pair. It has no relationship with real module's port IDs.

In the following example, QACT parses a GKV including SG1->SG2 firstly, in which the Splitter module's output port ID 5 is being used. The switch's output port ID 1 is used to route connection between SG1 and SG2. While parsing the second GKV SG1->SG3, in which the same Splitter module's output port 1 is being used. The switch will add a new output port ID 3 to route connection between SG1 and SG3.

D.6 Assign KV to switch port

While creating/updating a switch for one subgraph pair, typically, the source subgraph's SGKV will be assigned as the switch's input port KV, and the destination subgraph's SGKV will be assigned as the switch's output port KV. Then, an optimization will be applied to remove some duplicated KVs.

For a 1 to N type of switch, if its input port KV has a common part with all its output port KVs, the common part will be removed from all output port KVs. For a N to 1 type switch, if its output port KV has a common part with all its input port KVs, the common part will be removed from all input port KVs.



E References

E.1 Related documents

Document	
Qualcomm Technologies, Inc.	
Qualcomm Product Support Tool (QPST™) 2.7 User Guide	80-V1400-3
See *Qualcomm Unified Tools Service (QUTS) User Guide*	80-PG522-3
Using QACT to Customize ACDB Topologies	80-N7468-1
IIR PCM Filter	80-VR361-1
Application Note: Voltage-Current Feedback Speaker Protection V2.0 and Tuning Process	80-NT796-1
Presentation: Fluence v5 Acoustic Echo Cancelation Audio Tuning Training	80-NK880-2
Presentation: Fluence v5 Noise Suppression Audio Tuning Training	80-NK880-3
Presentation: Fluence v5.5 Broadside Noise Suppression Audio Tuning Training	80-NK880-4
Dynamic Range Control (DRC) Audio Tuning Guide	80-N2719-1
Multiband Dynamic Range Control (MBDRC)	80-N2719-2
Adaptive Input Gain Audio Tuning Guide	80-N2736-1
Application Note: Far-End Noise Suppression (FENS)	80-VU805-1
Fluence Pro Single-Mic and Multi-Mic Acoustic Echo Canceller Audio Tuning Guide	80-NB428-1
H2XML User Guide	80-VM407-19

E.2 Acronyms and terms

Term	Definition
ADB	Android Debug Bridge
EFS	Embedded File System
FSID	Feature Set ID
ANC	Active Noise Cancelation
AANC	Adaptive ANC
RTC	Real-Time Calibration
FTT	Fluence Tuning Tool
RTM	Real-Time Monitoring
SWP	Software Product