



Intel® INDE Visual Coding Framework User's Manual

Version: Beta-1 2016

LEGAL DISCLAIMER

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting [Intel's Web Site](#).

MPEG is an international standard for video compression/decompression promoted by ISO. Implementations of MPEG CODECs, or MPEG enabled platforms may require licenses from various entities, including Intel Corporation.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2007-2015, Intel Corporation. All Rights reserved.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Contents

1	Overview	6
2	Architecture	6
3	Installation	7
4	VCF Designer	8
4.1	Toolbar	10
4.1.1	File operations	10
4.1.2	Canvas edit control	11
4.1.3	Canvas edit mode control	11
4.1.4	Toolbar visibility control	11
4.1.5	Graph layout mode control.....	12
4.1.6	Graph Validation	12
4.1.7	Graph Execution control	12
4.1.8	Canvas zoom control.....	13
4.1.9	Guidance	13
4.1.10	Device link control.....	14
4.2	Left Panel	14
4.2.1	Designer Mode.....	14
4.2.2	Presets.....	15
4.3	Canvas	16
4.4	Right Panel	17
4.4.1	Graph Properties.....	17
4.4.2	Node Properties	18
4.5	Bottom Panel	21
4.5.1	Real-time metrics	21
4.5.2	Execution Status.....	22
4.5.3	History/Analysis	22

4.5.4	Dynamic Control.....	23
4.5.5	Stream Info.....	24
5	Device Link	25
5.1	Pre-requisites for Device Link	25
5.2	Using VCF Device Link	26
6	Tutorial.....	28
6.1	Exploring platform capabilities	28
6.2	Workload prototyping.....	29
6.3	Evaluate performance & behavior	34
6.4	Save workload	36
6.5	Integrate VCF workload with application	36
6.6	Build & Test.....	37
7	Default in/output folders.....	37
7.1	Content Input.....	37
7.2	Content Output.....	37
8	References	38

1 Overview

Visual Coding Framework (VCF) is a feature of Intel® INDE that allows developers to rapidly model and create code for media workloads. VCF provides a drag and drop tool, VCF Designer, to create a flow graph with audio or video along with end to end functions like file I/O to generate a powerful, high performance and scalable workloads. Workload can be run directly for VCF Designer, allowing quick exploration of platform features and performance. Once the workload is designed, VCF Designer can generate code (GraphML) for integration into a Windows* or Android* applications. By utilizing VCF, developers are not required to learn and use low-level SDKs or APIs.

VCF graph workloads (represented as GraphML files) are integrated into applications using the high level VCF API which exposes access to the cross-platform VCF run-time. VCF graphs are built with “nodes” which are fundamental functional execution blocks.

VCF graph workloads currently support a range of nodes, including:

- video encode, decode, processing
- audio encode, decode
- audio and video render, including audio resampling
- camera
- File I/O, including container multiplexing (muxing) and demuxing

The following chapters provide details about VCF architecture, workflow and the features of VCF Designer.

Please refer to separate VCF documentation for further details:

- VCF Release Notes – Current release limitations and known issues
- VCF SDK Manual – API reference manual

2 Architecture

The figure below provides a high level overview of the overall VCF architecture.

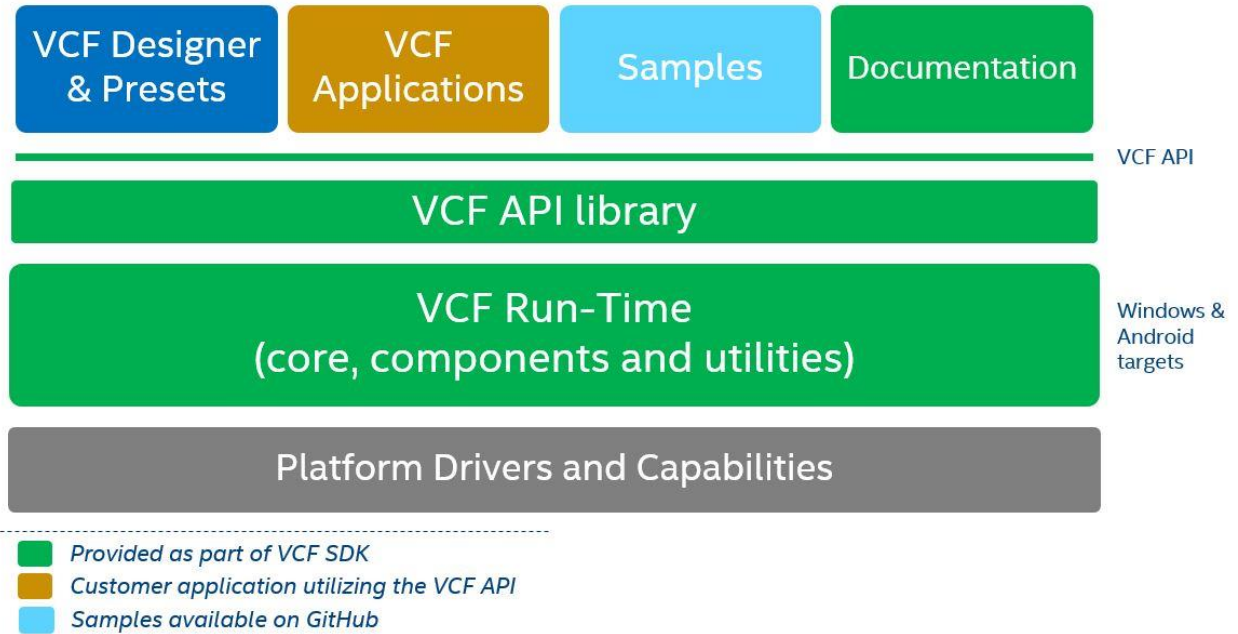


Figure 1: VCF Architecture

The VCF Designer component (colored “dark blue”) is the focus of the User’s Guide (this document). The components colored “green” in the figure above are part of the VCF SDK. VCF sample code (colored “light blue” above) is published on GitHub* repository: <https://github.com/INDExOS/visual-coding-framework>

Using the VCF Designer, users can design, prototype and benchmark graph workloads before starting integration into application code.

3 Installation

Installation of VCF results in the following folder hierarchy, located in <install-folder>. For instance: “C:\Intel\INDE\Visual_Coding_Framework_<version>”, using default install location.

Folder/File name	Description
<install-folder>	Root install folder
<install-folder>\vcf.exe	VCF Designer executable
<install-folder>\components\	VCF node implementations for Windows. Used for 64 bit prototyping.
<install-folder>\utils\	Common VCF utilities. Utilized by the VCF core and nodes.
<install-folder>\plugins\	VCF Designer User Interface (UI) functionality
<install-folder>\images\	VCF Designer image assets

<install-folder>\config\	VCF Designer configuration including node capabilities, parameter specification and descriptions
<install-folder>\platforms\ <install-folder>\docs\	VCF Designer UI framework functionality Contains this document and other VCF documentation such as the SDK reference manual
<install-folder>\presets\	Predefined set of presets to aid feature exploration
<install-folder>\templates\ <install-folder>\sdk	Templates for automatic graph generation used when exploring generated content The VCF SDK. Includes everything needed to build and deploy applications utilizing the graphs generated by the VCF Designer. Please refer to the SDK reference manual for details.
<install-folder>*.dll	The VCF Designer and the VCF Windows run-time depend on many features that are loaded dynamically. These are all represented by separate DLLs.
<install-folder>\VCFClientPlayer.apk	The VCF Android Client application. This application communicates with VCF Designer to control execution of workloads on a connected Android device.

Table 1: VCF Folder/File layout

Many of the provided presets depend on the Intel® INDE Content Media Package (<https://software.intel.com/sites/default/files/content.msi>). If not installed, when selecting a preset, user is prompted and suggested to install the package.

4 VCF Designer

Below is an annotated overview of the VCF Designer user interface, showcasing some of the key features of the tool.

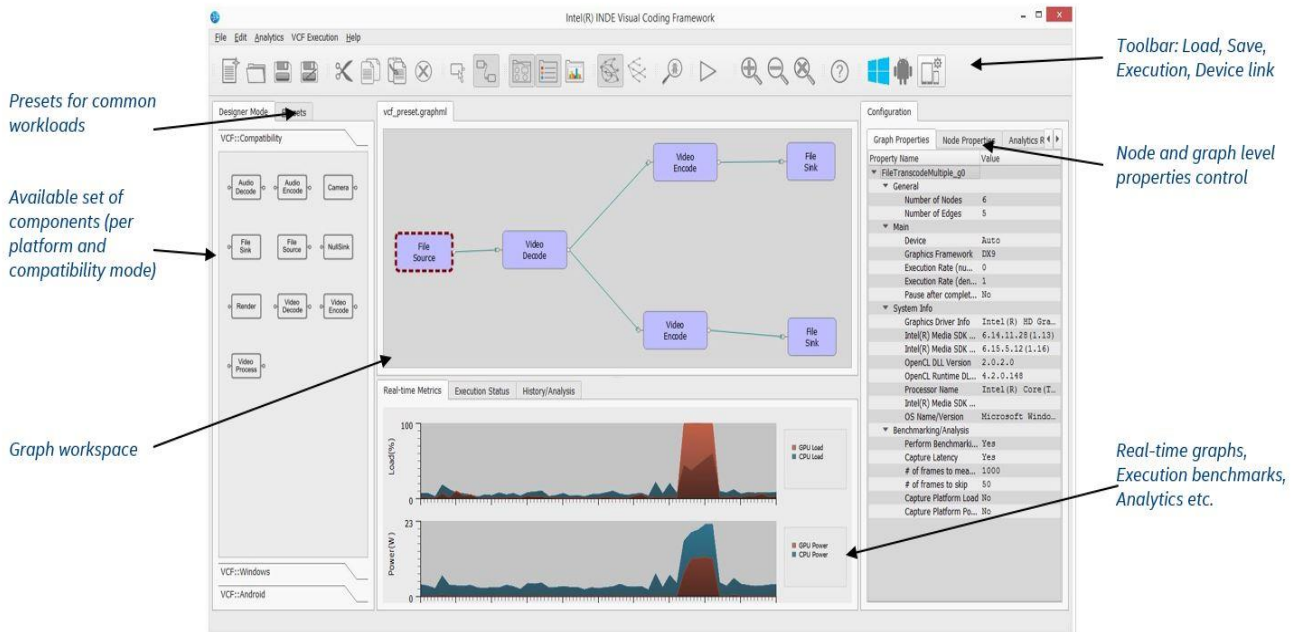


Figure 2: VCF Designer Feature Overview

The VCF Designer user interface is divided into four panels (Left, Right, Bottom and Canvas) and one toolbar. See figure below for an overview of the user interface layout. The following chapters provide details about the features hosted by each section of the user interface.

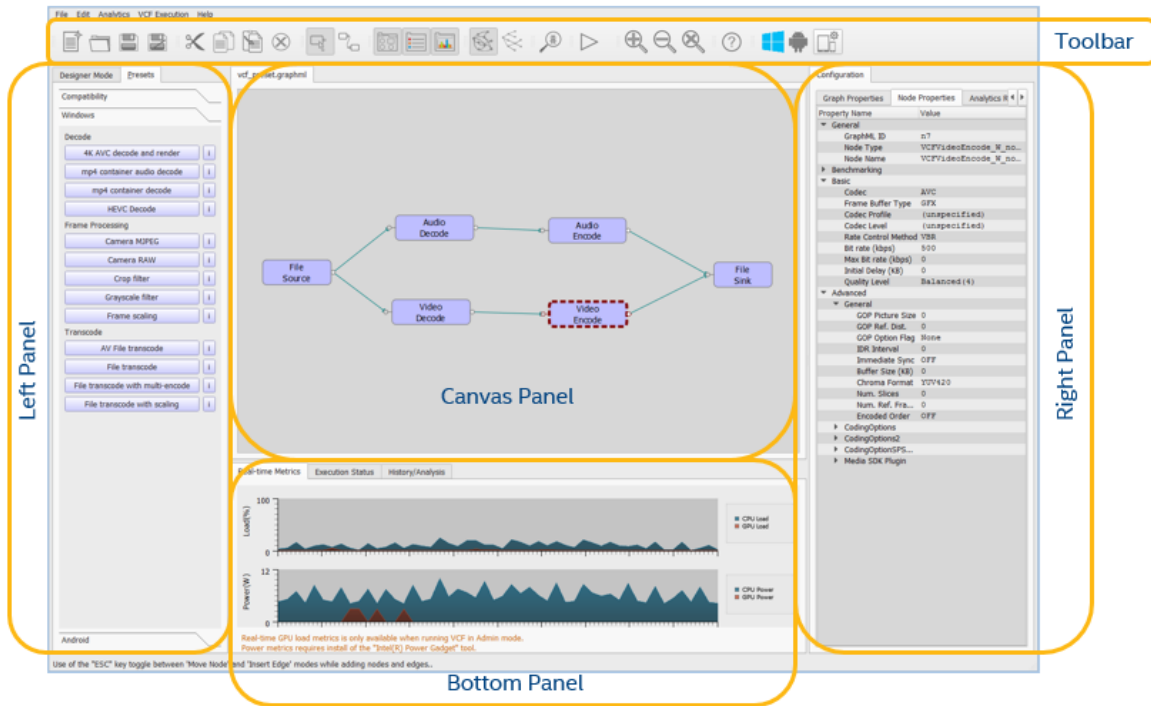


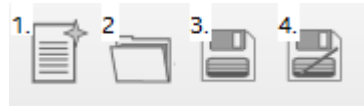
Figure 3: VCF Designer user interface layout

4.1 Toolbar

The VCF Designer toolbar features quick access to common operations such as graph loading, graph execution and design actions for editing graphs in the Canvas panel. The toolbar consists of several individual widgets, each containing a group of associated operations.

Like with most parts of the VCF Designer user interface, user can hover mouse over the toolbar elements to get brief tool tips on their meanings.

4.1.1 File operations



This toolbar group exposes the following file related operations:

1. **New Graph** (Shortcut key: Ctrl-N). Clears the graph canvas.
2. **Open Graph** (Shortcut key: Ctrl-O). Opens up a file selector dialog from which a GraphML file can be selected.
3. **Save Graph** using existing file name (Shortcut key: Ctrl-S).

4. **Save Graph using new file name.** Opens up a file selector dialog from which a new GraphML file can be selected.

4.1.2 Canvas edit control



This toolbar group exposes controls for compound editing of the graph in the Canvas panel. The following operations are available:

1. **Cut** selected elements of the graph (Shortcut key: Ctrl-X)
2. **Copy** selected elements of the graph (Shortcut key: Ctrl-C)
3. **Paste** copied or cut elements into graph (Shortcut key: Ctrl-V)
4. **Delete** selected elements from the graph (Shortcut key: Del)

4.1.3 Canvas edit mode control

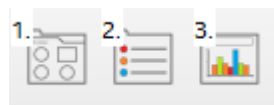


This toolbar group exposes controls for selecting active graph canvas editing mode. The following modes are available:

1. **Selection mode:** In this mode, any node or node edge connector may be selected and moved. Note that multiple graph elements may be selected by dragging the mouse to select a region of the canvas.
2. **Edge insertion mode:** In this mode new edges can be created between any nodes in the graph. Partial graph consistency validation is performed every time nodes are connected by an edge. If validation detects an issue, the user is prompted.

User may toggle between these modes by pressing the Escape (Esc) shortcut key.

4.1.4 Toolbar visibility control



This toolbar group allows user to show or hide (toggle) the panels of the user interface. The following operations are available:

1. **Show/Hide Left Panel**
2. **Show/Hide Right Panel**
3. **Show/Hide Bottom Panel**

Note that some panels may appear automatically based on specific VCF Designer events.

When VCF Designer starts, the Left and Right panel are visible. The Canvas panel is always visible.

4.1.5 Graph layout mode control



This toolbar group permits user to select the desired graph layout mode. The following modes are available:

1. **Custom layout mode:** Locations of nodes and edges are displayed as specified by the user.
2. **Hierarchical layout mode:** Location of nodes and edges are analyzed and the layout is organized in a hierarchical manner.

4.1.6 Graph Validation

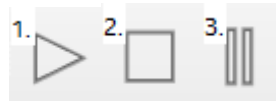


This control group only has one member, with the purpose of performing static rules checks on graph. The operation detects static issues in the graph currently being designed on the Canvas. Issues such as dangling nodes, missing sink or source, or node incompatibilities are detected.

Results of the rule check is displayed in the Right panel (“Analytics Report” tab). If the Right panel is hidden, it will be opened to display the rule check results. The get details about identified issues, click on any of the issues listed in the report.

Graph validation may also be invoked by using Shortcut key: Ctrl-Shift-R

4.1.7 Graph Execution control



This toolbar group exposes controls for graph execution. The following operations are available:

1. **Play:** The Play operation instructs the active VCF run-time to execute the graph. Before graph is executed the graph is validated to ensure that it can be processed by the VCF run-time. While graph is executing the Status tab in the Bottom panel is updated dynamically. (Shortcut key: Ctrl-Shift-P)
2. **Stop:** The Stop operation instructs the active VCF run-time to stop graph execution. Upon completed execution the Status tab in the Bottom panel is updated. (Shortcut key: Ctrl-Shift-T)
3. **Pause:** The Pause operation instructs the active VCF run-time to pause graph execution. The pause operation acts in toggle fashion. So to un-pause a paused just press the “Pause” button again. (Shortcut key: Ctrl-Shift-O)

The availability of the operations in this group depend on the current execution context. Also note that, while a graph is executing, new graph may not be loaded or current graph edited.

4.1.8 Canvas zoom control



This toolbar group exposes controls for graph canvas zoom level. The following operations are available:

1. **Zoom in**
2. **Zoom out**
3. **Reset zoom** level to default level

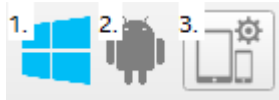
4.1.9 Guidance



This toolbar group contains a single button which activates the “What’s this” mode. Using this mode users may click on any UI section to get details about its functionality.

Guidance mode may also be activated by Shortcut key: Shift-F1

4.1.10 Device link control



This toolbar control group enables user to select the desired VCF run-time on which to execute workloads.

1. **Windows VCF run-time state:** Colored icon indicates that Windows VCF run-time is active
2. **Android VCF run-time state:** Colored icon indicates that Android VCF run-time is active
3. **Device link configuration:** Clicking this button opens the VCF run-time selection dialog

Please refer to the “Device Link” chapter for details about this capability.

4.2 Left Panel

The Left Panel has a two purposes, exposed as individual tabs, described in the following sections.

4.2.1 Designer Mode

The Designer Mode tab displays all the available nodes which can be used to design graphs. The tab is divided into sections, expressing the three available node groups:

1. **Compatibility:** Default mode ensuring cross-OS and cross-platform parity with limited set of functional capabilities and parameters
2. **Windows:** Nodes from this section expose capabilities and properties specific for Windows systems
3. **Android:** Nodes from this section expose capabilities and properties specific for Android systems

To insert a node into the graph, drag it from the Left panel to the Canvas. Nodes can also be inserted into canvas by clicking on the desired node then clicking again on the Canvas.

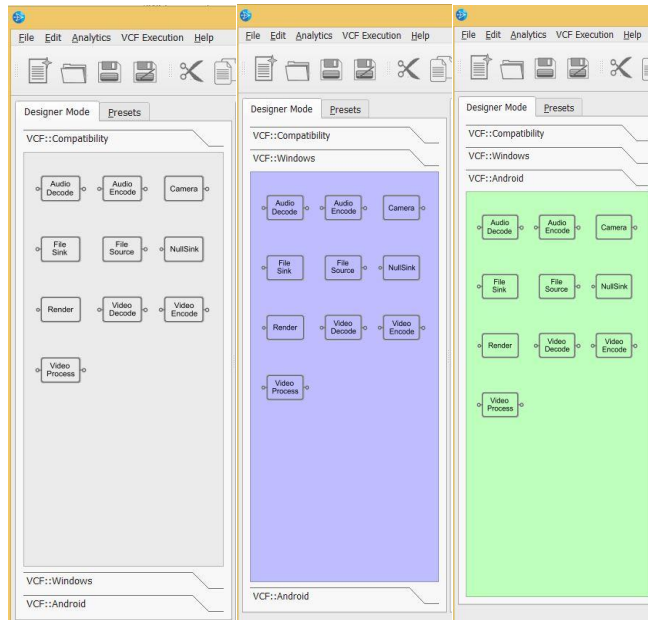


Figure 4: Designer Mode layout

Some important usage notes:

- “Compatibility” nodes may be combined with OS specific nodes.
- OS specific nodes may not be mixed. For example, a graph using both Android and Windows nodes is not allowed.
- A graph with OS specific nodes cannot be executed on other OS than for which it was designed.

4.2.2 Presets

The Presets tab displays all available presets, divided into the same three categories as the Designer Mode tab: Compatibility, Windows and Android. Presets are essentially pre-designed graphs, designed with the purpose of showcasing a specific workloads supported by VCF.

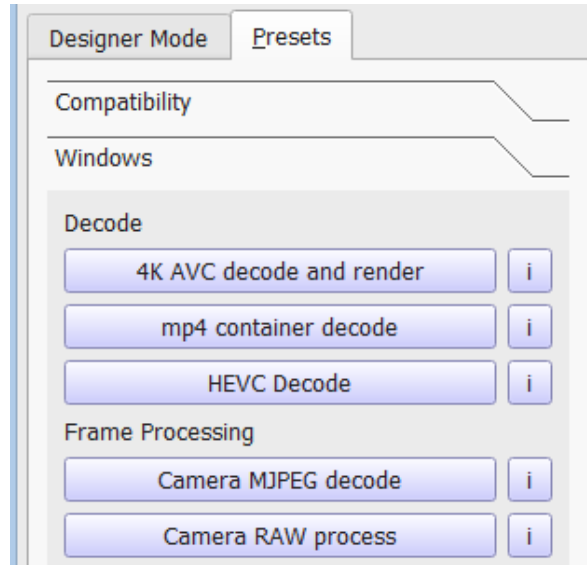


Figure 5: Preset selection dialog

By selecting one of the presets, the corresponding graph is loaded. Details about the specific presets are also available by pressing the “i” button next to the preset.

Note that many presets depend on file based input content. Such presets refer to files which are part of the Intel® INDE Content Media package. This package is provided as a separate installer, located here: <https://software.intel.com/sites/default/files/content.msi>. If preset with file based input content is selected, and this package has not yet been installed, VCF will prompt the user.

4.3 Canvas

The Canvas section of the user interface is where graph workloads are designed. To construct a graph, drag nodes from the Left panel to the Canvas and use the “Edge insertion mode” to connect nodes together.

The “Canvas Edit Control” operations available in the toolbar may be used to quickly design more complex graphs.

Keep in mind that all graphs must have at least one source node and one sink node. Also, graphs must not have any dangling nodes and must adhere to node input and output type compatibilities. The VCF Designer tool utilizes multiple techniques to ensure that graphs are valid, such as validation before graph execution, before saving graph and manually by pressing the “Graph Validation” button in the toolbar. The tool also performs edge compatibility validation every time two nodes are joined.

4.4 Right Panel

The Right panel has one main tab named “Configuration”, which name indicates the primary purpose of the panel. Using this panel users can control configuration for each node (the “Node Properties” sub-tab) part of the graph and also the overall configuration for the graph (the “Graph Properties” sub-tab). There is also a third tab, named “Analytics Report”, which provides information about results from static rules checks performed on the graph.

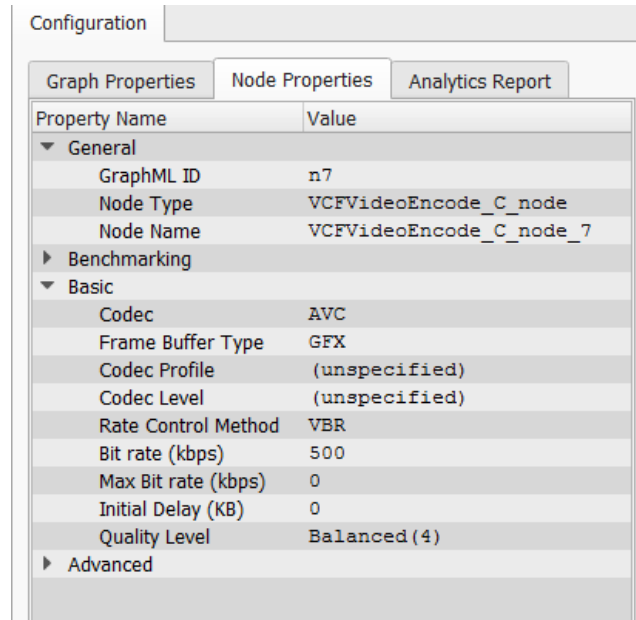


Figure 6: Node Properties dialog

4.4.1 Graph Properties

Graph properties applies to the graph as a whole. Some examples of graph level properties are:

- “Execution Device”: CPU, GPU or Auto (GPU used, if not available, use CPU). GPU means use Intel® Processor Graphics or hardware-based acceleration.
- “Execution Rate”: The rate at which the workload executes. Default value is 0 which means maximum speed. User may also specify execution at specific rate using the Numerator (N) and Denominator (D) property parts.
- Benchmarking: Properties for graph performance benchmarking with metrics such as CPU/GPU load and latency.

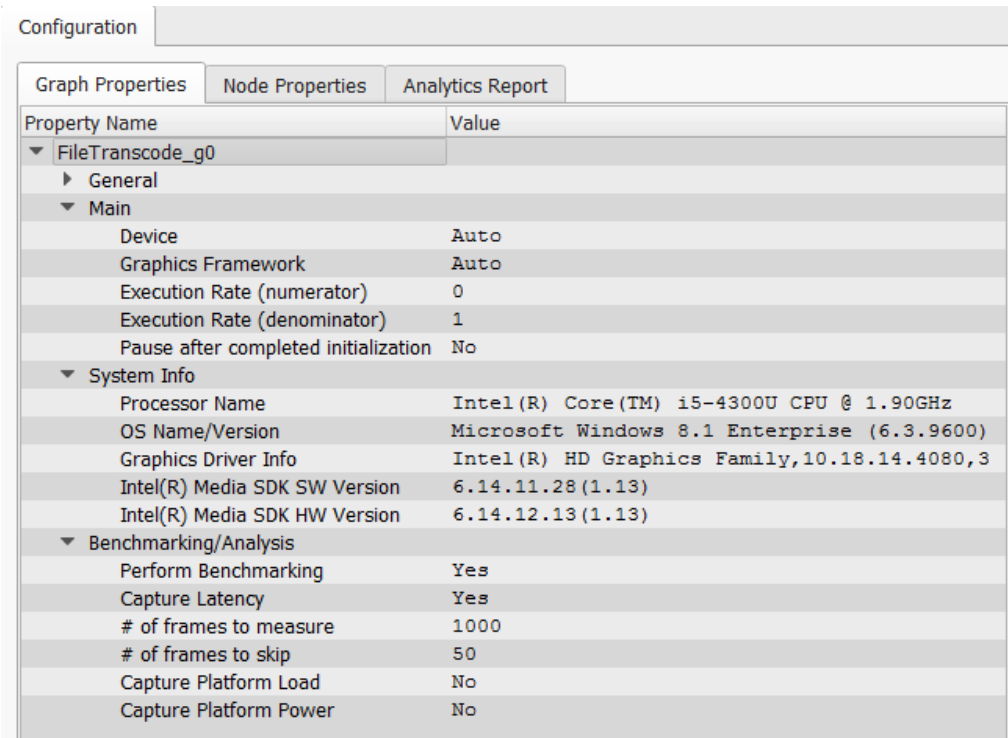


Figure 7: Graph Properties dialog

The Graph properties tab also has a sub category of uneditable properties, named “System Info”, which indicates high level system information about the system on which the VCF runtime is running (either locally on Windows* or remotely on Android*).

4.4.2 Node Properties

Click on the desired node in the Canvas to display properties associated with a node. The properties are divided into several groups with the first group named “General”, present for all node types. The “General” group contains a few unmodifiable properties such as the unique node instance name.

The “General” group is followed by node specific property groups. The group names and properties depend on the node type. In the example screen capture below, a “Video Encode” node is selected which has three groups named “Benchmarking”, “Basic” and “Advanced”.

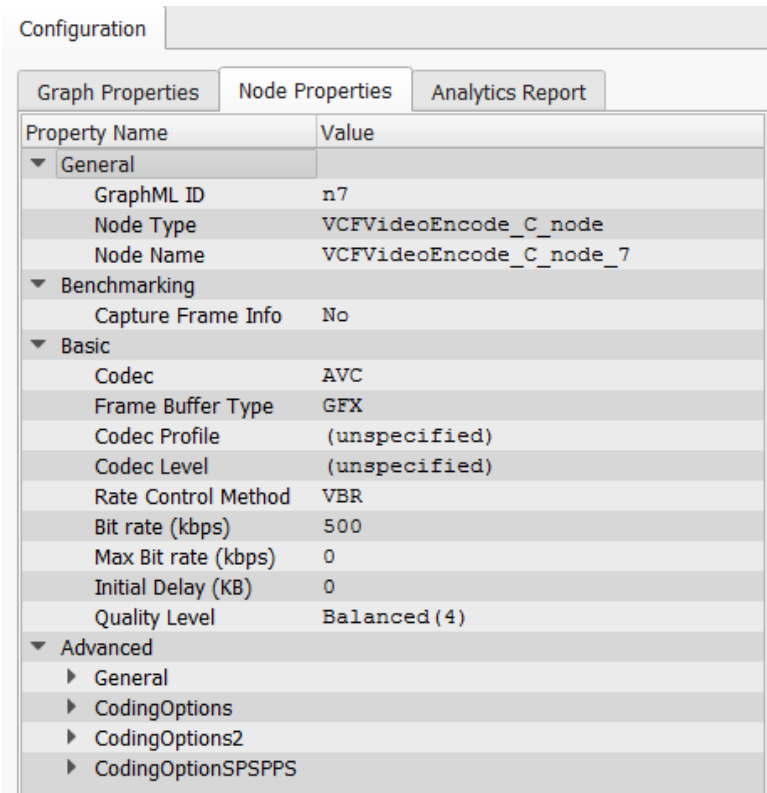


Figure 8: Node properties dialog

Node properties are often divided into basic and advanced groups to simplify navigation and to cater to both novice and expert users. Also note that some groups (such as the “Advanced” tab in the example above) has further group nesting.

Some groups have a leading property, named “Enable”, which is used to activate the properties in the group.

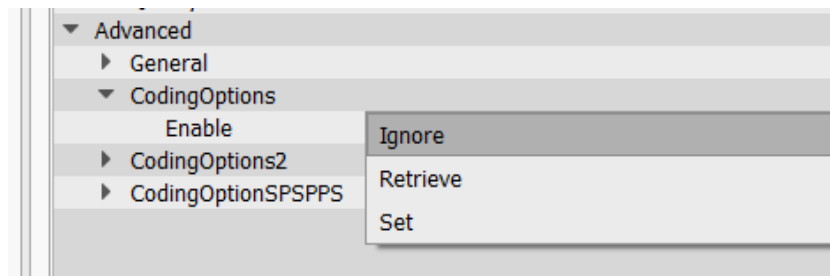


Figure 9: Property group activation

As illustrated in the above example, users may select either “Ignore” (default), “Set” (means that the properties in the group will be used) or “Retrieve” (means that the node is queried for default property values).


After executing a graph, an additional column is displayed for some nodes, listing the actual property value used during execution. Note that the actual value is only be displayed if it differs from the configured value. See example below.

The screenshot shows a configuration window with three tabs: 'Graph Properties', 'Node Properties', and 'Analytics Report'. The 'Node Properties' tab is active, displaying a table with three columns: 'Property Name', 'Value', and 'Actual Value'. The table is organized into sections: 'General', 'Benchmarking', and 'Basic'. The 'Basic' section contains several properties, with the last four rows highlighted by an orange box. These rows show that the 'Actual Value' differs from the 'Value' for 'Codec Profile', 'Codec Level', 'Max Bit rate (kbps)', and 'Initial Delay (KB)'.

Property Name	Value	Actual Value
▼ General		
GraphML ID	n7	
Node Type	VCFVideoEncode...	
Node Name	VCFVideoEncode...	
▼ Benchmarking		
Capture Frame Info	No	
▼ Basic		
Codec	AVC	
Frame Buffer Type	GFX	
Codec Profile	(unspecified)	AVC_HIGH
Codec Level	(unspecified)	AVC_3.1
Rate Control Method	VBR	
Bit rate (kbps)	500	
Max Bit rate (kbps)	0	750
Initial Delay (KB)	0	93
Quality Level	Balanced (4)	

Figure 10: Display of actual property values

Below are a few other things to note about the process of changing property values:

- If a file selection property is selected, the user has the option of opening up a file selector dialog by clicking on the folder icon  next to the manual file name input field.
- Some properties provide a list of valid options (multi-selection) and some properties require numeric input within a specified range. Multi-selection properties options may impact the availability of other options due to node specific property dependencies
- Verbose details are displayed for each property by hovering the mouse pointer over the property. This action displays a tooltip info box containing verbose info, unique property identifier and value range.
- Node properties and graphs may not be modified while a graph workload is being executed.

4.5 Bottom Panel

The Bottom panel of the VCF Designer user interface is the host of many features, all organized in separate tabs.

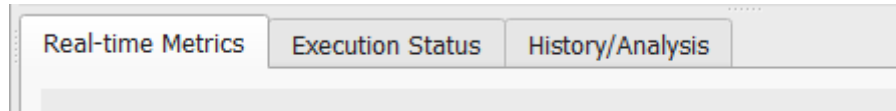


Figure 11: Tabulated features in Bottom panel

Below is the list of currently available features:

1. Real-time metrics
2. Execution Status
3. History/Analysis
4. Dynamic Control
5. Stream info

Each of the above are described in the following sections.

4.5.1 Real-time metrics

The Real-time metrics tab shows live metrics captured by the currently active VCF run-time. Metrics are presented in two graphs: Load and Power, both including metrics for both CPU and GPU.

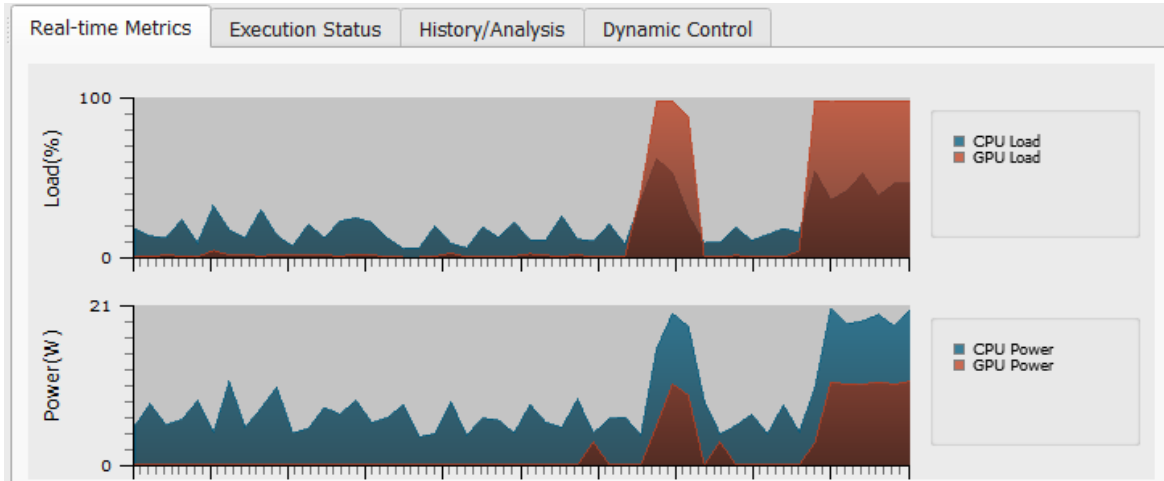


Figure 12: Real-time metrics dialog

Some important notes about real-time metrics availability:

- For Windows* GPU load metrics, VCF Designer must be executed in Windows* administration mode.

- For Windows* power metrics, the installation of Intel® Power Gadget is required. (<http://software.intel.com/en-us/articles/intel-power-gadget-20>). Note that VCF Designer only supports 64-bit version of Power Gadget

4.5.2 Execution Status

The Execution Status tab displays details about executed graph workloads including benchmarks and execution log. The “Progress”, “Execution Log” and “Status” field are updated dynamically as the workload is being executed.

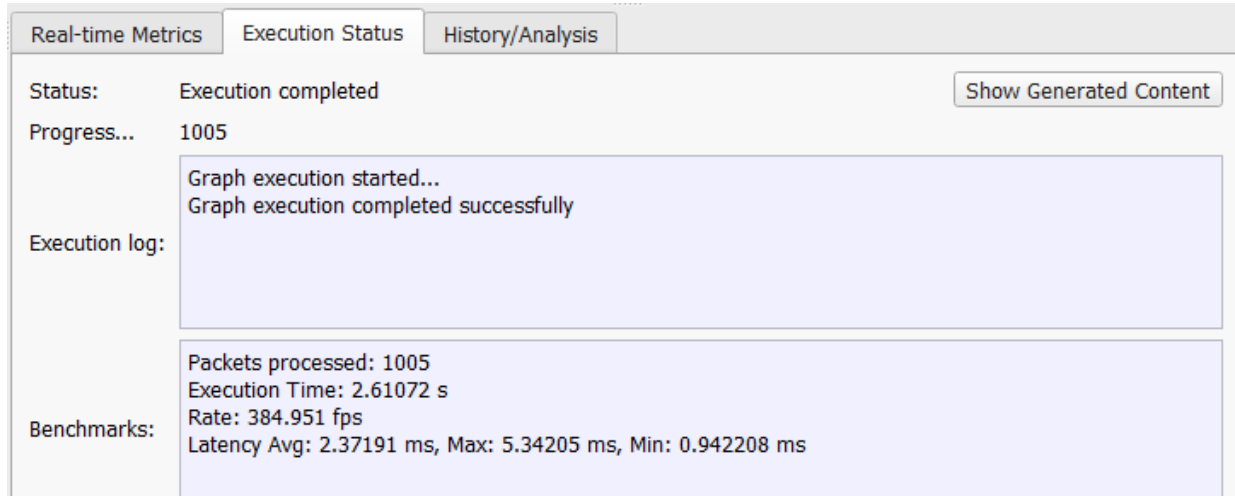


Figure 13: Execution status dialog

The level of details displayed in the tab depend on the benchmarking options selected from the “Graph Properties” tab, part of the Right panel. For instance, via the benchmarking options, user may select to capture average power and load during the time the workload executes, or to turn off benchmarking completely.

The “Execution Log” log section conveys overall info about key events and also shows any errors which may have been encountered.

The Execution Status tab also features the option to explore any generated content artifacts from a graph workload execution by clicking the “Show Generated Content” button. Note that this button is only available for graphs which output content to files.

4.5.3 History/Analysis

The History/Analysis tab allows users to compare the performance from multiple workload executions. Benchmarks for all executed workloads are recorded and displayed in this tab, including the option to annotate, delete or revisit workloads.

To compare a selected set of workload benchmarks, click on the radio button next to the desired metric name in the table header. This action results in a bar chart being displayed below the table of recorded benchmark data.

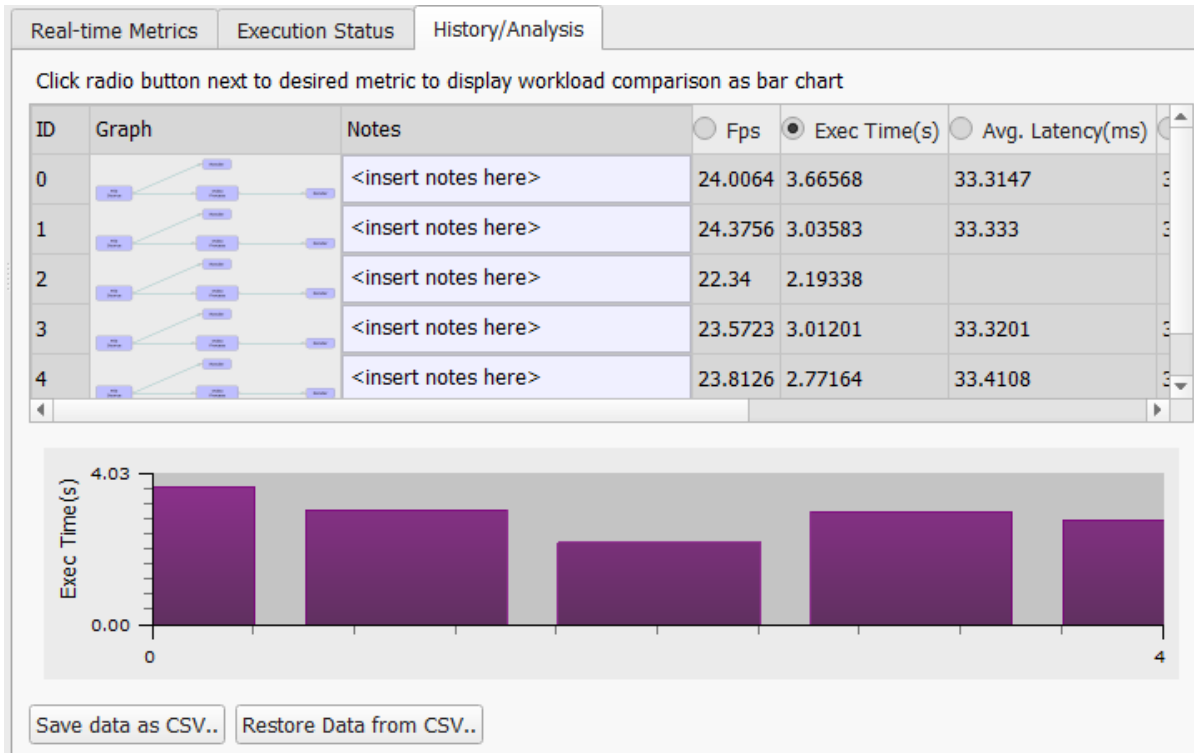


Figure 14: History/Analysis dialog

Collected records may also be saved or restored to/from CSV files.

4.5.4 Dynamic Control

The Dynamic Control tab offers the ability to control certain graph and node properties dynamically, while the graph is being executed. For instance, the Video Process node Brightness property may be modified using slider or number input as showcased in the screen capture below.

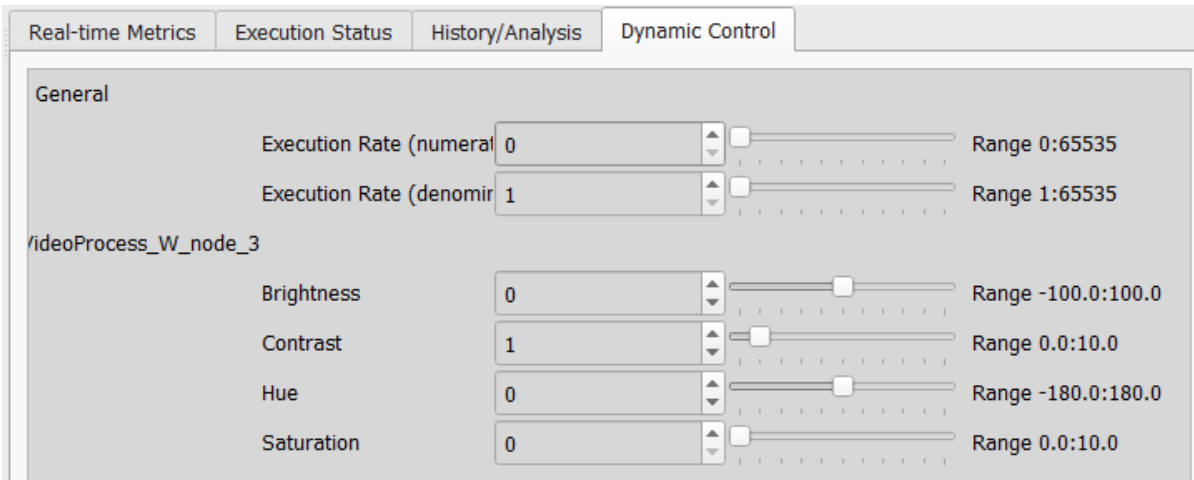


Figure 15: Dynamic property control dialog

Note that the Dynamic Control tab is only displayed while the workload is being executed.

4.5.5 Stream Info

The Stream Info tab is associated with the Video Encode node and is displayed for the case when the user has selected to turn on benchmarking for this Video Encode node type, by setting the “Capture Frame Info” property, under “Benchmarking” group, to “Yes”.

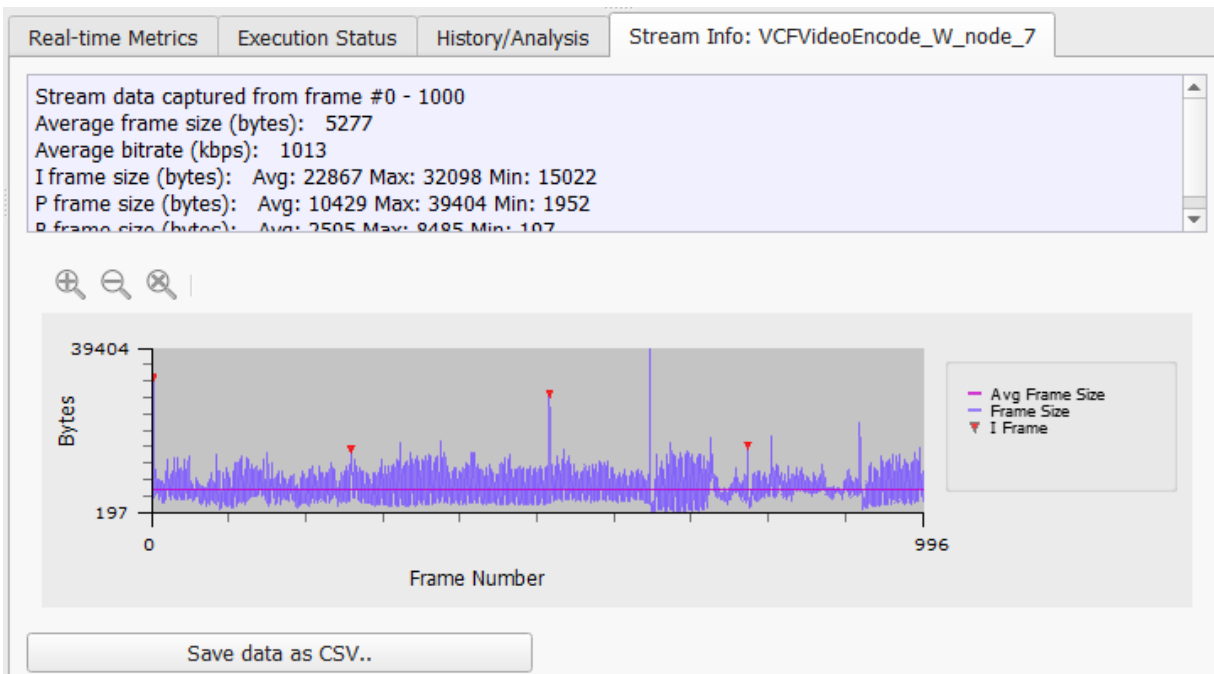


Figure 16: Stream Info dialog

When enabled, the tab is displayed after successful graph execution, providing detailed information about generated stream, such as average bit rate, average frame size per frame type and more. A chart is also displayed showing the size of each generated frame.

Users may also save the captured stream data as a CSV file for import into external data manipulation tool.

5 Device Link

VCF Designer not only supports local workload execution, but users can also run workloads on remote devices to perform real-time workload analysis using Android* device link capability. Currently this feature support is available for Android* KitKat* (4.4) and Android* Lollipop* (5.x) on Intel® Atom™ processors with Intel® Processor Graphics.

5.1 Pre-requisites for Device Link

The device link feature requires the following items:

- Android* Debug Bridge (ADB) (which is a part of Android SDK), must be installed and path added to the machine environment variables.
 - Android* SDK - <https://developer.android.com/sdk/index.html>
 - More info about ADB - <http://developer.android.com/tools/help/adb.html>
- Developer Options must be enabled on the Android* Device (<http://developer.android.com/tools/device.html#device-developer-options>).
- The ADB USB Driver must be installed on the host to enable and connect device via USB. (<http://developer.android.com/tools/extras/oem-usb.html#InstallingDriver>)

In order to verify that the installation is done correctly, connect the Android Device to the Host through USB. Then run “adb devices” from command line to see the Android Device listed. If all the items are correctly installed, user should see the devices under “List of devices attached” as shown below:

```
C:\>adb devices
List of devices attached
503013637412    device
```


Figure 177: “adb devices” command output example

5.2 Using VCF Device Link

After VCF Designer is started, notice the VCF Device Link section in the toolbar. The Windows local host is selected by default.



In order to initiate a connection to an Android device, perform the following steps:

1. Press the  button. Indicated by tooltip “Select to Configure VCF Runtime”
2. Wait for VCF to query available devices.
3. Once the devices are queried, the “Select a Device” dialog appears

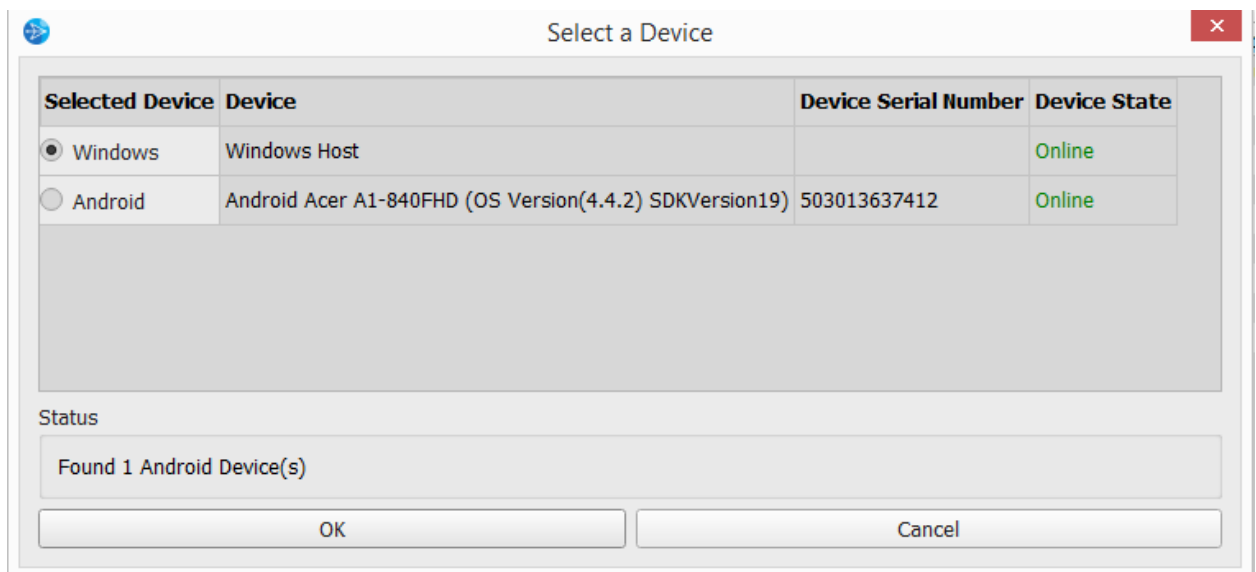


Figure 188: Select a Device dialog

4. User can select any device that is in “Online” state. Once the device is selected, Press OK.
5. Wait for VCF to establish connection to the Device.

6. After successful connection the VCFClientPlayer application is started on the connected device

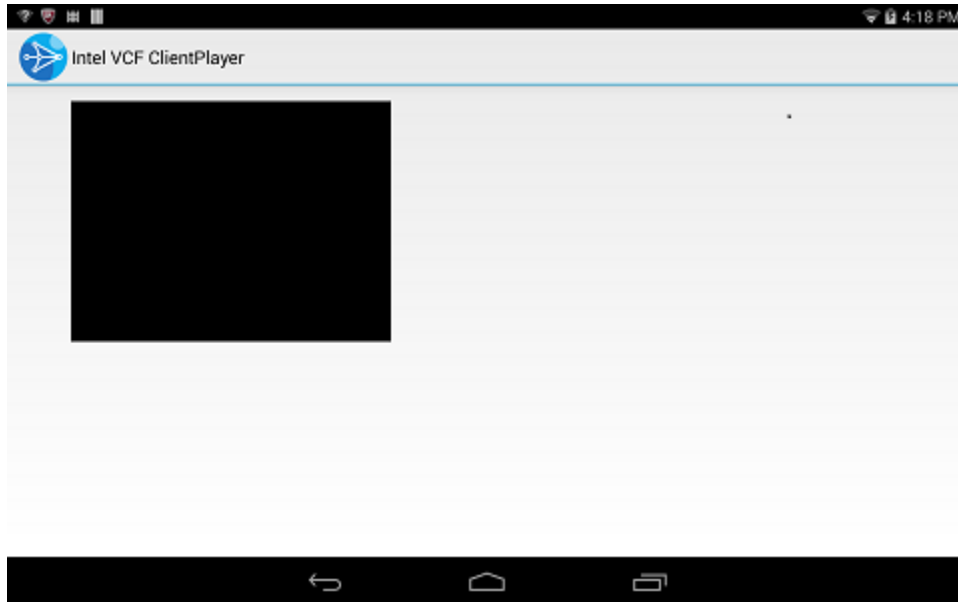



Figure 199: VCFClientPlayer Running on Android Device

7. At this point VCF Designer is connected to the device. Users can run Compatibility or Android Presets on their device as specified in Section 4.2.2 (Presets). Users can also create their own graphs in the designer mode as specified in Section 4.2.1 (Designer Mode).
8. Workloads are executed by pressing the Run button(), which will invoke execution of the workload using the VCFClientPlayer on the Android* device. The status, real-time metrics, history and more will be updated in the VCF Designer as the workload is executed.
 - a. If a file is referenced in the FileSource node make sure it is on the device (in folder /sdcard/vcf/) before executing the workload. Files may be copied manually using adb.
9. In order to switch back VCF workload execution to the local host, follow steps 1 thru 4 above, selecting Windows Host. Switching back to the Windows Host run-time, will close the VCFClientPlayer on the device.

6 Tutorial

Below figure illustrates a complete VCF workflow, from workload exploration to application deployment.

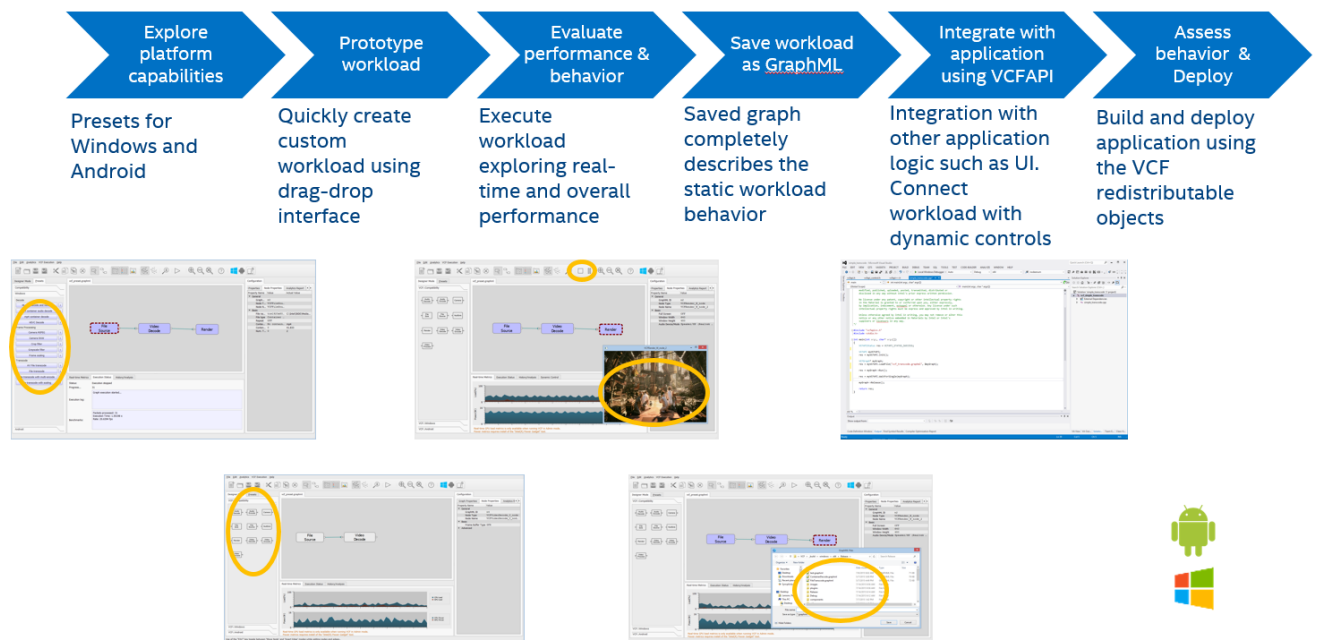


Figure 20: VCF workflow overview

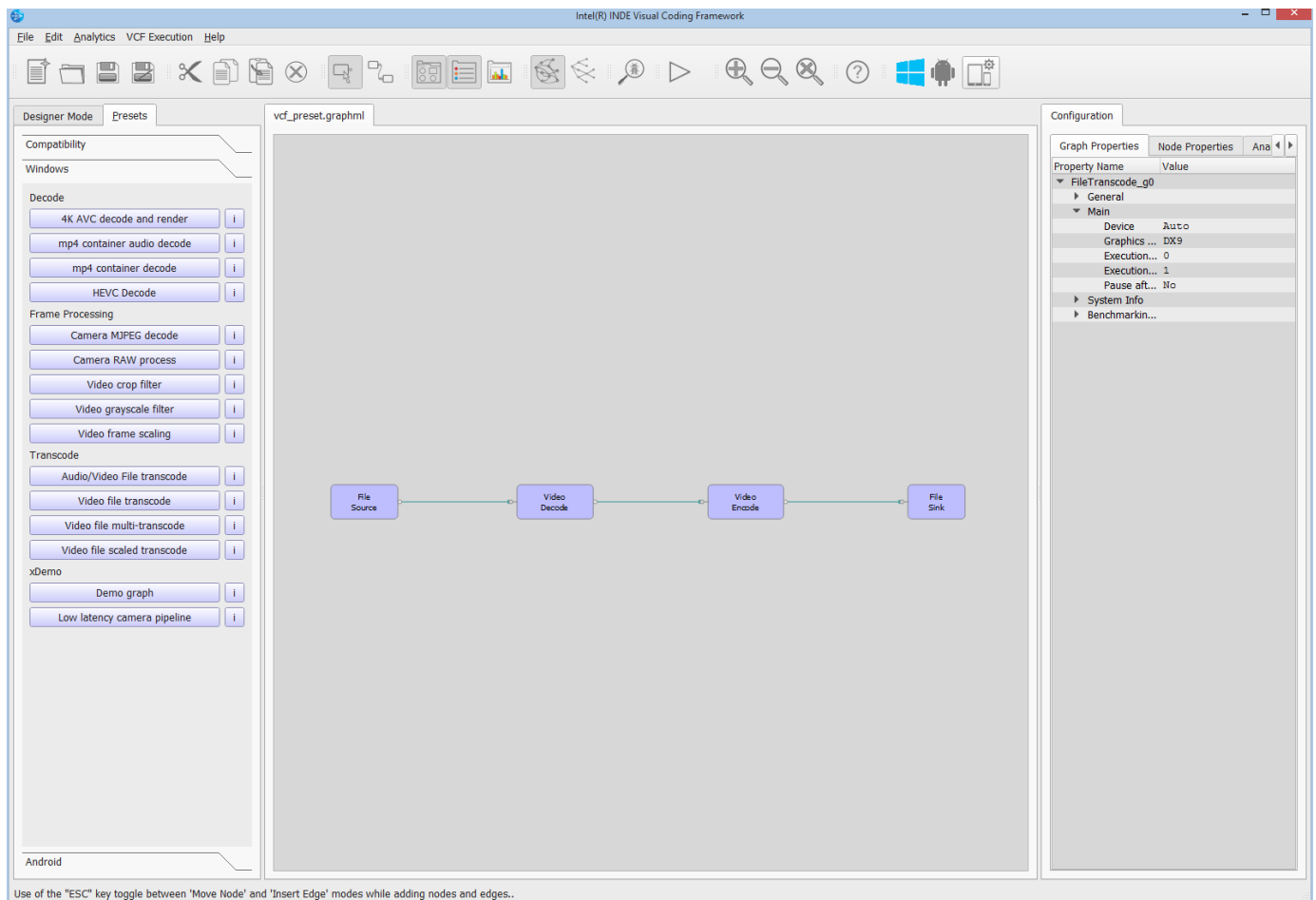
Compared to traditional usage of lower level APIs and components, VCF provides a visual and flexible framework for rapid application creation and deployment. By utilizing VCF developers can save many months of development time and invest less resources, due to not having to spend time and resources mastering low level library or SDK APIs.

This chapter provide a step by step guide taking the user through all the steps of the workflow required to build a Windows* media transcoding application.

6.1 Exploring platform capabilities

When the VCF Designer tool is first launched, the user is presented with a blank canvas, on which VCF graph workloads may be designed. To the left of the canvas there is a panel, which by default displays a list of VCF graph presets.

Navigate to the left panel and click on the “Windows” subsection at the bottom of the tab to display a list of available VCF presets designed to showcase VCF capabilities for Windows* OS. Locate and click on the preset named “Video File Transcode”. This will load the preset graph into the canvas. To view details about the selected preset click on the “I” button next to the preset button.



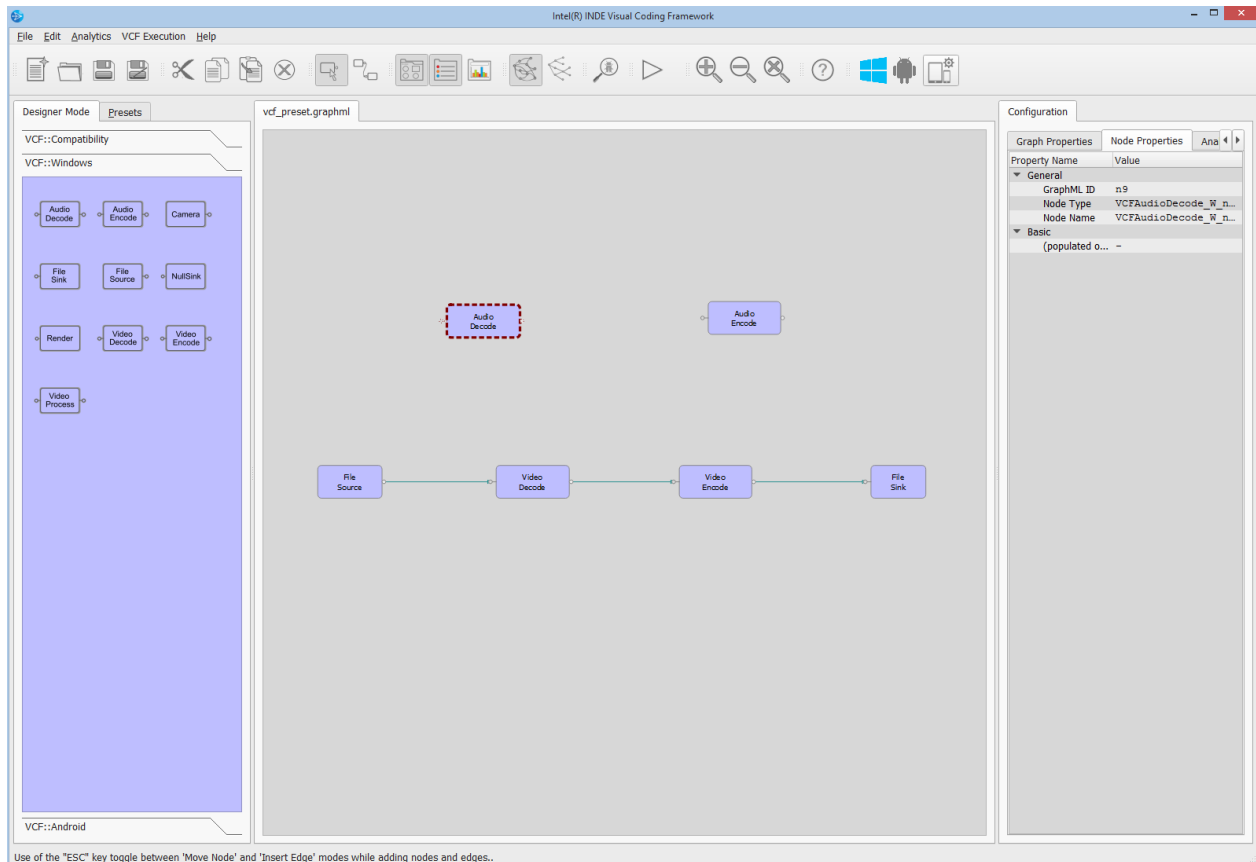
To explore the default configuration for a node in the graph click on the desired node on the canvas, then navigate to the panel to the right of the canvas. This panel lists all available node properties and their default values. The properties are organized in groups which can be expanded and collapsed. There are also graph level properties which are accessed by selecting the “Graph Properties” tab in the same panel.


This workload can be executed to explore behavior and gather benchmarks, but first let’s explore how the graph can be modified to create custom workload.

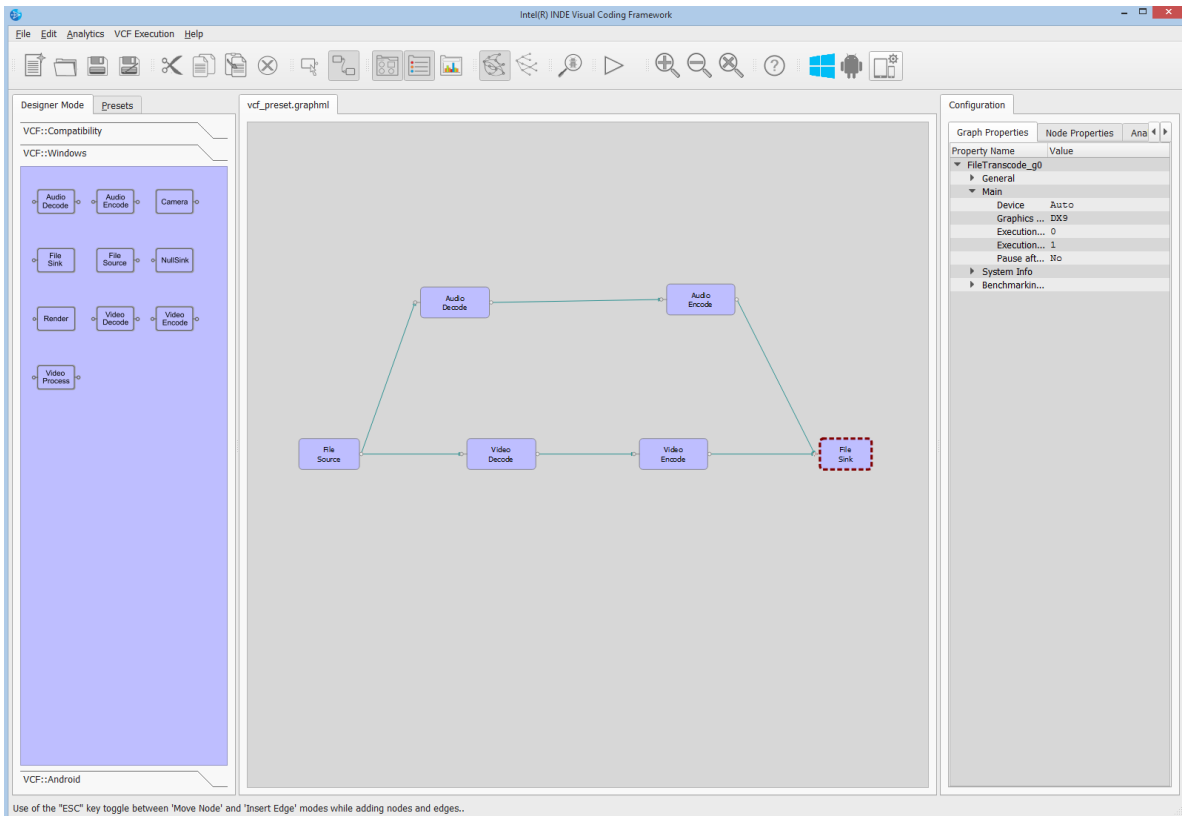
6.2 Workload prototyping


Activate the graph design mode on the left panel of the user interface by clicking on the “Designer Mode” tab. In this tutorial we will extend the graph to add support for audio processing, video frame scaling, and media container handling.


First, let's add the required audio nodes by dragging an "Audio Decode" and an "Audio Encode" node to the canvas from the left panel.

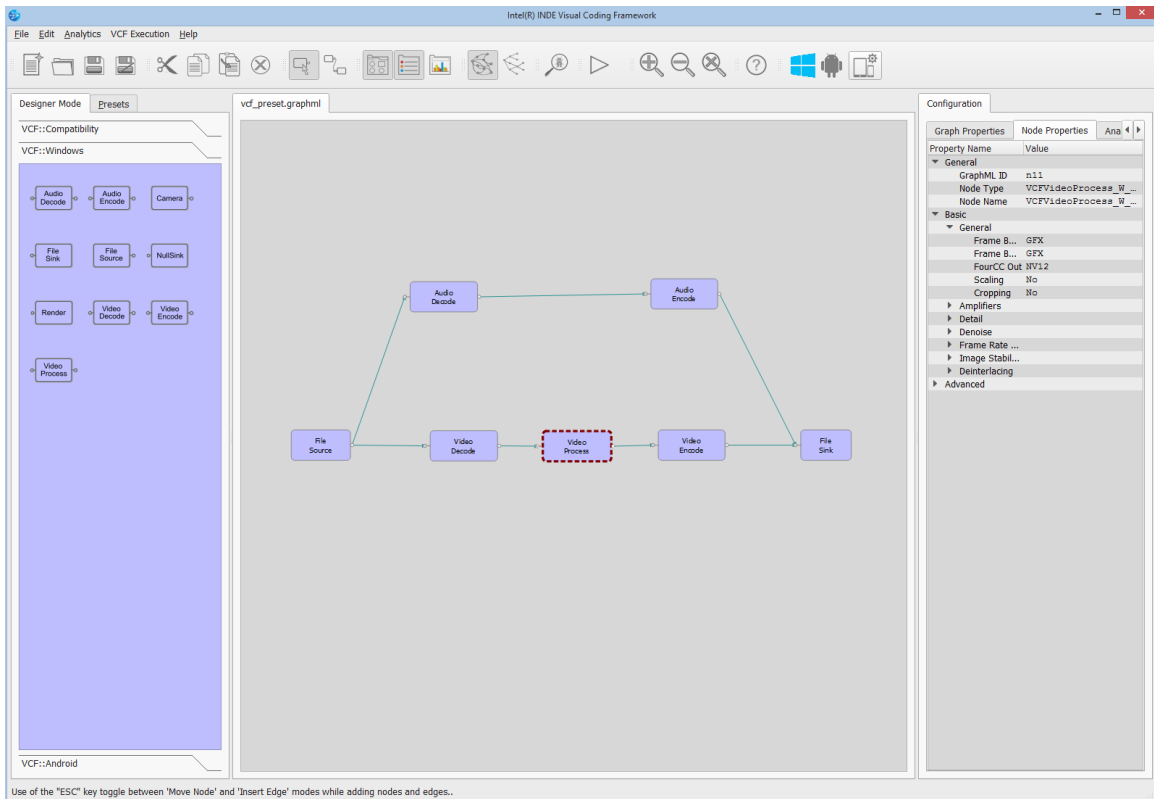


Then connect the "File Source" node with the "Audio Decode" node by first enabling the "Edge insertion mode" () from the toolbar (note that the selection modes may be toggled by pressing the "Escape" key). Connect the nodes by dragging an edge between the "File Source" output to the "Audio Decode" input. Perform the same action to connect the "Audio Decode" node with the "Audio Encode" node, and the "Audio Encode" node with the "File Sink" node.




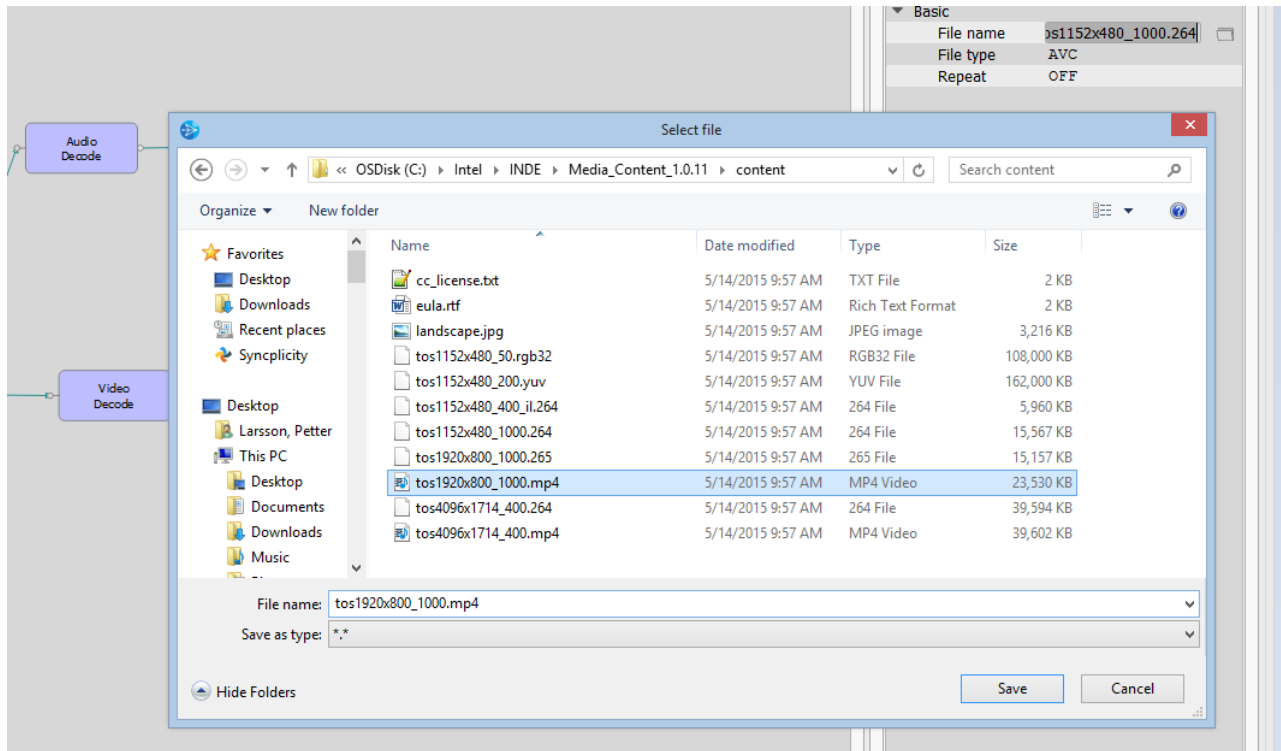
Next step is to add a “Video Process” node with purpose of changing the resolution of the transcoded video stream. First erase the edge connection between the “Video Decode” and the “Video Encode” node by selecting the connection, then clicking the Delete button (), or pressing the “Del” key.

Remember to switch to the node move mode () first. Next drag the “Video Process” node on to the canvas, placing it between the “Video Decode” and the “Video Encode” node. Note that the nodes may have to be moved slightly to make room for the new node. Next, connect an edge between the “Video Decode” node with the “Video Process” node using the “Edge insertion mode”. Also connect the “Video Process” node with the “Video Encode” node.



A few more changes are required to make the graph valid, allowing us to execute the workload.

First we change the input file from and default “AVC” elementary stream video file to an “mp4” media container file. This is achieved by clicking the “File Source” node, then selecting value next to the “File name” property in the right panel. Click on the folder icon () to open up a file selection dialog in which the mp4 file is selected. Note that the Intel® INDE Media Content package (<https://software.intel.com/sites/default/files/content.msi>) contains various content, including a sample .mp4 file (including both video and audio). Click “Save” on the file selector dialog to commit the file selection.



Since the input file type was changed, the “File type” property must be modified. To do this, click on the current value (“AVC”) to bring up multi-value selector, then select the “Container...” value. Now select the “File Sink” node and then modify the “Multiplexing option” property to the value “mp4”. This ensures that the output is packaged as a media container in the “mp4” format.

Property Name	Value
▼ General	
GraphML ID	n5
Node Type	VCFFileSource_W_node
Node Name	VCFFileSource_W_node_5
▼ Basic	
File name	C:/Intel/INDE/Media_Content_1.0.11/c...
File type	Container (mp4/mpeg-ts/mkv)
Repeat	OFF

Property Name	Value
▼ General	
GraphML ID	n8
Node Type	VCFFileSink_W_node
Node Name	VCFFileSink_W_node_8
▼ Basic	
File name	output
RAW video options	Write NV12 as I420/IYUV
Multiplexing options	mp4

Finally, the “Video Process” node is modified to set the frame scaling properties. Do this by selecting the multi-value property named “Scaling”, setting it to “Yes”. Pressing “Enter” key, then modifying the value of the properties “Width” and “Height” to 640 and 480 respectively.

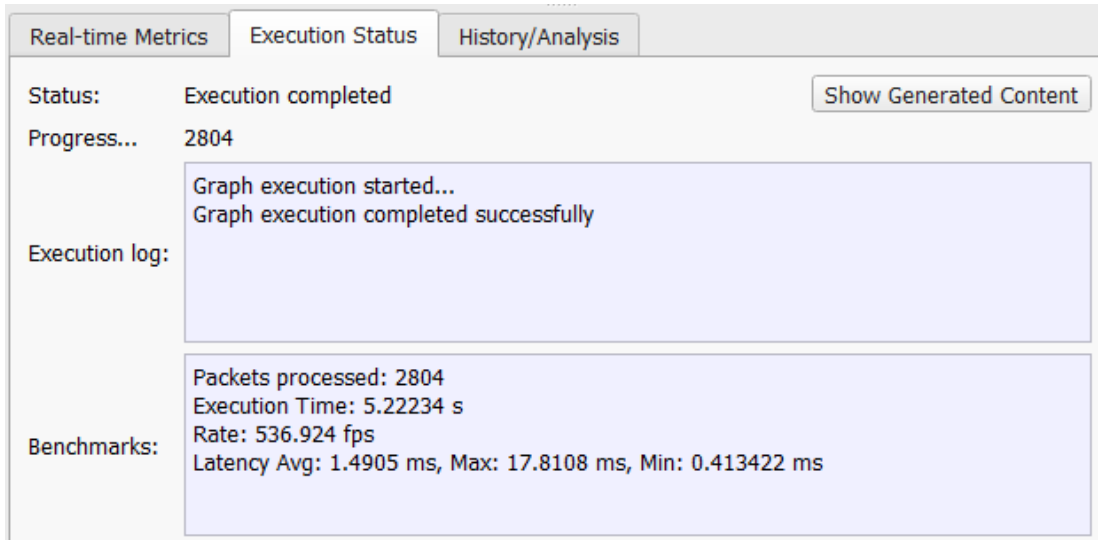
Source Out	AVI
Scaling	Yes
Width	640
Height	480

The graph is now ready to be executed to explore behavior and performance.

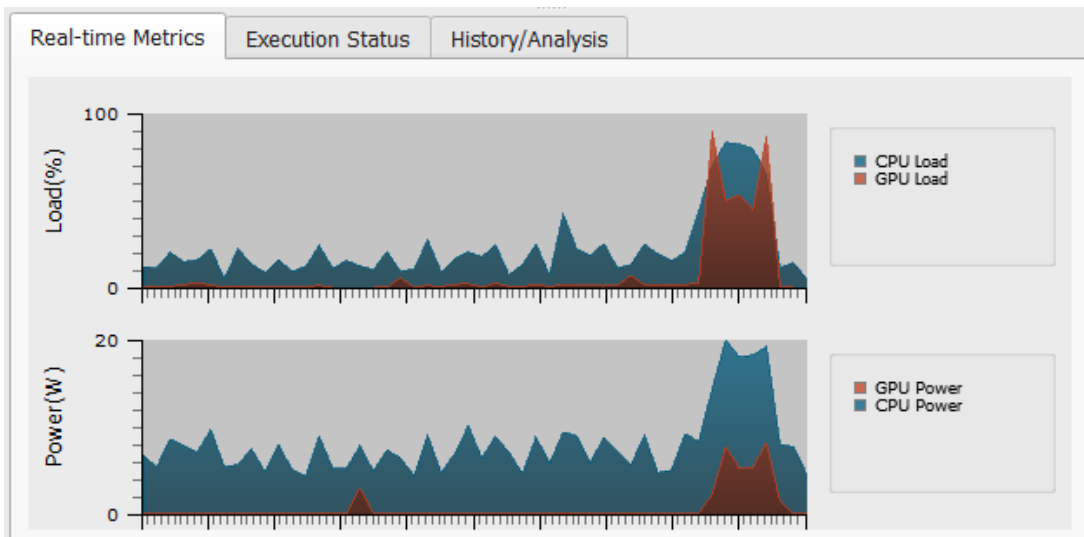
6.3 Evaluate performance & behavior

To execute the graph currently residing on the canvas, click on the Play icon (▶) in the toolbar. This operation also invoke graph validation, which will abort execution and display a list of issues in the right panel “Analytics Report” tab if issues are found.

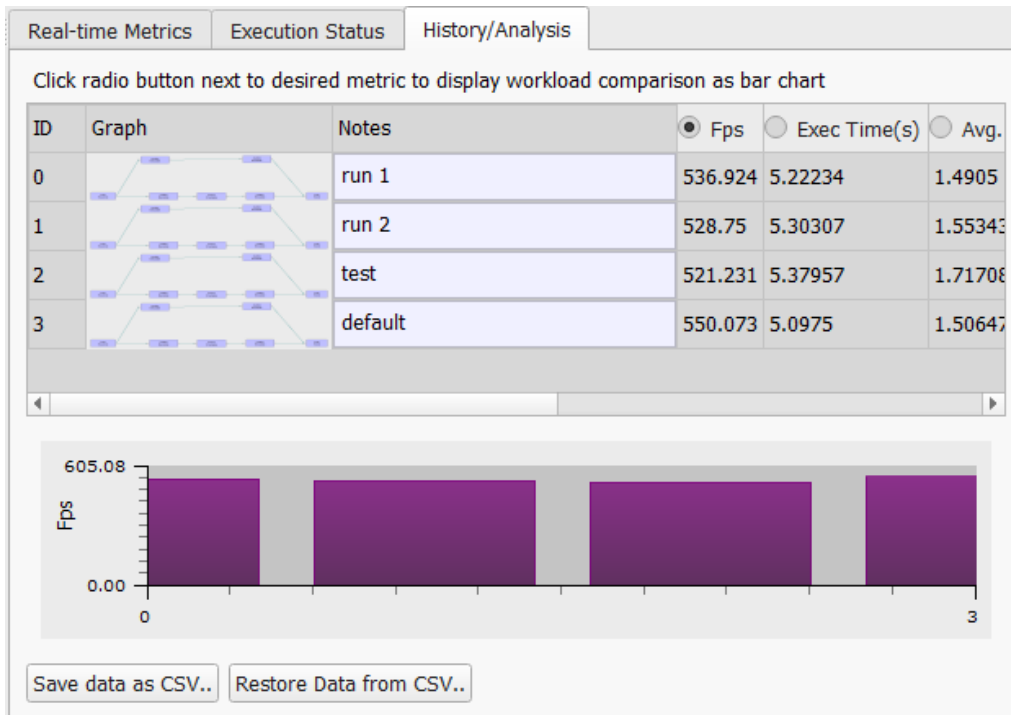
At the moment the graph starts executing the bottom panel “Execution Status” tab is automatically opened. This tab displays key information about the execution progress, overall status, log and benchmarks. After the graph has completed execution the generated file content can be explored by pressing the “Show Generated Content” button.



The bottom panel also features the ability to display real-time metrics for CPU/GPU load and power. To explore these metrics click on the “Real-time Metrics” tab.




All benchmarks for executed workloads are recorded, which allows VCF Designer to compare performance across multiple workload runs. This feature is exposed in the “History/Analysis” tab where benchmarks can be explored as bar charts, annotated and saved/restored.



Also note that execution may be stopped or paused at any time during execution by pressing the Stop or Pause icons in the toolbar.

6.4 Save workload

After exploring node capabilities and performance the VCF graph can be saved as a VCF GraphML file by clicking on the “Save As...” icon () in the toolbar. This action brings up a file selector dialog in which the user may select the file name and location of the VCF GraphML file.

The VCF GraphML file represent the complete static workload description of the workload, including all graph level properties and node properties. The GraphML is the generated “source code”. Note that it is not recommended to edit the GraphML file manually, instead open up VCF Designer and load the file to make modifications to the properties and/or layout.

6.5 Integrate VCF workload with application

To integrate the saved VCF GraphML file with an application the VCF API interface are used. The VCF API is part of the VCF SDK which is described in detail as part of the “VCF SDK Manual” document. Please refer to the “Programming Guide” chapter in this document for a quick overview on how a GraphML file is loaded and executed using the API.

6.6 Build & Test

Building an application using the VCF API, GraphML and content requires Microsoft* Visual Studio (refer to VCF Release Notes for supported versions).

A simple starting point for an application using the VCF API is found in the VCF source code repository on GitHub (<https://github.com/INDExOS/visual-coding-framework>). Follow the below steps to create application project and build solution:

1. Download the “simple_transcode” sample project and open it up in Visual Studio.
2. Locate the line in the code with the text “res = myVCFAPI.LoadFile(“vcf_transcode.graphml”, &myGraph);” in the “simple_transcode.cpp” file. Replace the name of the GraphML file saved earlier with the default file name (“vcf_transcode.graphml”).
3. Set the “VCF_SDK_ROOT” environment variable to point to the VCF SDK folder (default location is “C:\Intel\INDE\Visual_Coding_Framework_<version>\sdk\vcfapi”).
4. Build the solution for the desired target, e.g. x64/Debug.
5. Copy the saved GraphML file and the input “mp4” media content to the same folder as the executable you built in the previous step
6. Run the built application from Visual Studio or from the command line

After completing these basic application integration steps, users may explore using other parts of the VCF API to dynamically control graph execution and properties.

7 Default in/output folders

7.1 Content Input

If file path is not specified, VCF will attempt to load the content from the Intel® INDE Content Media package folder (for example: C:\Intel\INDE\Media_Content_<version>\content). If this folder does not exist or the content file referenced is not present, the load will fail.

Intel® INDE Content Media package is located here:

<https://software.intel.com/sites/default/files/content.msi>

7.2 Content Output

If file path is not specified VCF will attempt to save the file in the default VCF output location:

<user’s document folder>/Intel_VCF/

8 References

- Visual Coding Framework on the Web: <https://software.intel.com/en-us/visual-coding-framework>
- Visual Coding Framework Sample code on GitHub: <https://github.com/INDExOS/visual-coding-framework>