



Scene-auto Usage Guide

Issue **00B01**

Date **2019-01-18**

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd. 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon (Shanghai) Technologies Co., Ltd.

Trademarks and Permissions



HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road, Bantian,
Longgang District,
Shenzhen 518129 P. R. China

Website: <http://www.hisilicon.com/en/>

Email: support@hisilicon.com



About This Document

Purpose

The scene-auto module adjusts the image and encoding parameters according to the luminance change in the environment and the user-defined bit rate, so as to optimize the image display effect.

Related Version

The following table lists the product versions related to this document.

Product Name	Version
Hi3516C	V500
Hi3516D	V300
Hi3516A	V300



Intended Audience




This document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
	Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

Symbol	Description
	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.
	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
	Calls attention to important information, best practices and tips. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Issue 00B01 (2019-01-18)

This issue is the first draft release.



Contents

About This Document.....	ii
1 Introduction.....	1
1.1 Scene-auto Sample Overview	1
1.1.1 Scene-auto Modules.....	1
1.1.2 Module Block Diagram.....	3
1.2 Scene-auto Sample Module Overview	3
1.2.1 Import of the Scene-auto Parameters	4
1.2.2 Main Scene-auto Framework	4
1.2.3 Scene-auto Adjustment Module	5
1.3 File Structure	6
1.4 Important Concepts	6
2 Development Guide	8
2.1 Usage Procedure.....	8
2.1.1 Overview.....	8
2.1.2 Running the Sensorini File.....	8
2.1.3 Obtaining and Importing Scene-auto and Media Channel Parameters.....	9
2.1.4 Enabling the Scene-auto Function	10
2.1.5 Switching Scene-auto Configuration	10
2.1.6 .ini Parser	11
2.1.7 Scene-auto Configuration Parameters.....	12
2.1.8 Parameters for Configuring the Media Channel.....	12
2.2 Running the Sample with a New Sensor	13
2.3 Running the Sample with Parameters Added or Deleted.....	14
2.4 Typical Pipeline Configuration Solutions.....	17
2.4.1 Preview and Video Recording with a Single Sensor	17
2.4.2 Preview and Snapshot with a Single Sensor	18
2.4.3 Preview and Video Recording with Multiple Sensors.....	19
2.4.4 WDR Preview and Video Recording with a Single Sensor.....	19
3 API Reference	21
4 Data Structure	26
5 Error Code.....	31



6 Parameter Description.....	32
6.1 ISP Parameters	32
6.1.1 AE Parameters.....	33
6.1.2 Global CAC Parameters.....	34
6.1.3 Saturation Parameters	35
6.1.4 LDCI Parameters	35
6.1.5 Dehaze Parameters	36
6.1.6 Mesh Shading Parameters	37
6.1.7 WDRExposureAttr Parameters	37
6.1.8 FSWDR Parameters	39
6.1.9 Gamma Parameters	39
6.1.10 DRC Parameters.....	39
6.1.11 3DNR Parameters.....	42
6.1.12 Sharpen Parameters.....	43
6.1.13 Demosaic Parameters	43
6.1.14 BayerNR Parameters.....	43
6.1.15 Detail Parameters	44



Figures

Figure 1-1 Module enabling and disabling in the .ini file	2
Figure 1-2 Scene-auto sample framework.....	3
Figure 1-3 Obtaining scene-auto sample parameters.....	4
Figure 1-4 Scene-auto framework code.....	5
Figure 1-5 Scene-auto adjustment module	5
Figure 1-6 File organization of the scene-auto module	6
Figure 1-7 Pipeline for preview and snapshot with a single sensor.....	7
Figure 2-1 Process of using the scene-auto module	8
Figure 2-2 Process of using the scene-auto module	9
Figure 2-3 .ini configuration management files required by cfgaccess	11
Figure 2-4 .ini configuration file for the dynamic AE parameters.....	12
Figure 2-5 .ini configuration file of the media channel parameters.....	13
Figure 2-6 Contents in the param folder of the sensor	14
Figure 2-7 Adding a parameter to the scene-auto configuration items.....	15
Figure 2-8 Adding a parameter to the scene-auto header file for value assignment	15
Figure 2-9 Adding a parameter to the scene-auto source file for value assignment	16
Figure 2-10 Adding a parameter to the scene-auto source file for value obtaining	17
Figure 2-11 Pipeline for preview and video recording with a single sensor.....	17
Figure 2-12 Pipeline for preview and snapshot with a single sensor.....	18
Figure 2-13 Pipeline for preview and video recording with multiple sensors	19
Figure 2-14 Pipeline for preview and video recording in 2-to-1 WDR mode with a single sensor.....	20



Tables

Table 2-1 Parameter configuration A of HI_SCENE_MODE_S	18
Table 2-2 Parameter configuration B of HI_SCENE_MODE_S.....	18
Table 2-3 Parameter configuration C of HI_SCENE_MODE_S.....	19
Table 2-4 Parameter configuration D of HI_SCENE_MODE_S.....	20
Table 5-1 Error codes provided by the scene-auto module.....	31
Table 6-1 Static parameters of the AE module	33
Table 6-2 Dynamic parameters of the AE module.....	34
Table 6-3 Static parameters of the global CAC module	34
Table 6-4 Static parameters of the saturation module.....	35
Table 6-5 Static parameters of the LDCI module	36
Table 6-6 Static parameters of the Dehaze module.....	36
Table 6-7 Dynamic parameters of the Dehaze module	36
Table 6-8 Static parameters of the Mesh Shading module.....	37
Table 6-9 Dynamic parameters of the Mesh Shading module	37
Table 6-10 Static parameters of the WDRExposureAttr module.....	38
Table 6-11 Gamma parameters	39
Table 6-12 DRC parameters in linear mode	40
Table 6-13 DRC parameters in WDR mode	41
Table 6-14 Parameters of the sharpen module.....	43
Table 6-15 Parameters of the demosaic module	43
Table 6-16 BayerNR Parameters	44
Table 6-17 Parameters of the detail module	44



1 Introduction

Cameras are influenced by the changes in external scenes during video shooting. In different scenes, the image sensor processor (ISP) in the chip uses different parameters to achieve the optimal effect of the collected image in the current scene.

1.1 Scene-auto Sample Overview

1.1.1 Scene-auto Modules

For the earlier scene-auto sample, when multiple sensors are adapted, each sensor needs to maintain a set of scene-auto code because the parameter type and parameter value to be adjusted are different. Therefore, the code quantity is large. If the parameters added to a sensor are configured to another sensor, other scene-auto codes need to be adjusted as well, increasing maintenance difficulties.

Hi3516C V500 and Hi3519A V100 share the same scene-auto architecture. Different sensors share one set of parameter value assignment code. Currently, the module switches are used to configure the ISP parameters for different modes and different sensors.

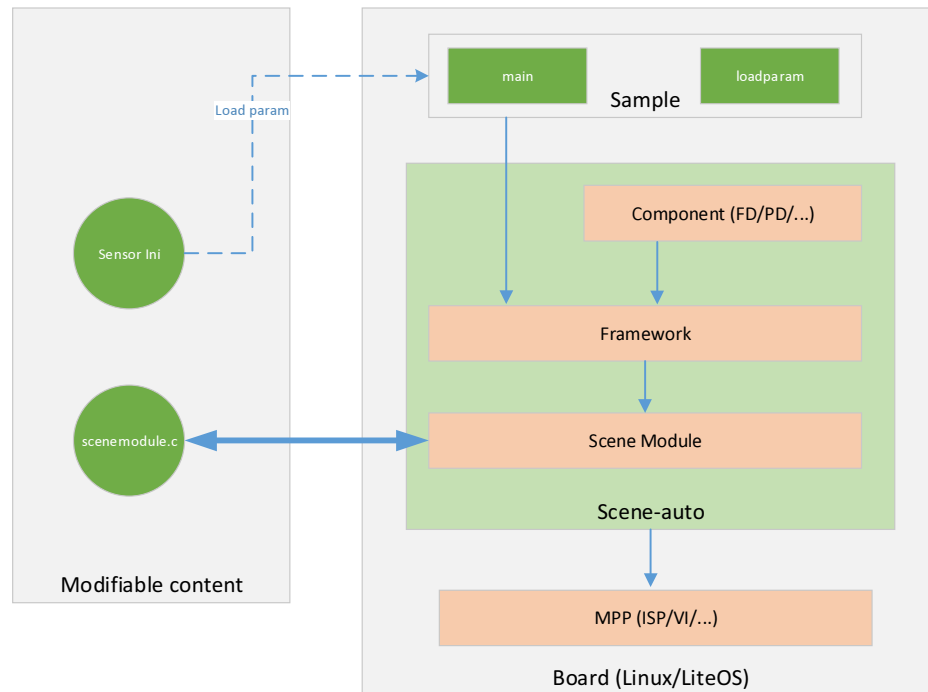
As shown in [Figure 1-1](#), **0** indicates that a module is disabled and **1** indicates that a module is enabled. This parameter is not configured in the corresponding API of the module.

**Figure 1-1** Module enabling and disabling in the .ini file

```
;;;;;;;;;module state;;;;;;;;;
[module_state]
bStaticAE                = "1"
bStaticAWB                = "0"
bStaticAWBEx             = "0"
bStaticCCM                = "1"
bStaticAERouteEx         = "1"
bStaticGlobalCac         = "0"
bStaticWdrExposure       = "0"
bStaticDehaze             = "0"
bStaticLdci               = "0"
bAeWeightTab             = "1"
bStaticSaturation         = "1"
bStaticShading            = "0"
bDynamicThreeDNR         = "1"
bDynamicGamma            = "1"
bDynamicShading          = "0"
bDynamicLdci             = "1"
bDynamicDehaze           = "0"
bDynamicFSWDR            = "0"
bStaticSharpen           = "1"
bStaticDemosaic          = "1"
bDynamicBayernr          = "1"
bStaticDetail            = "1"
bDynamicDis              = "0"
bStaticDrc               = "1"
bDynamicLinearDrc        = "1"
bDynamicThreadDrc        = "0"
```

1.1.2 Module Block Diagram

Figure 1-2 Scene-auto sample framework



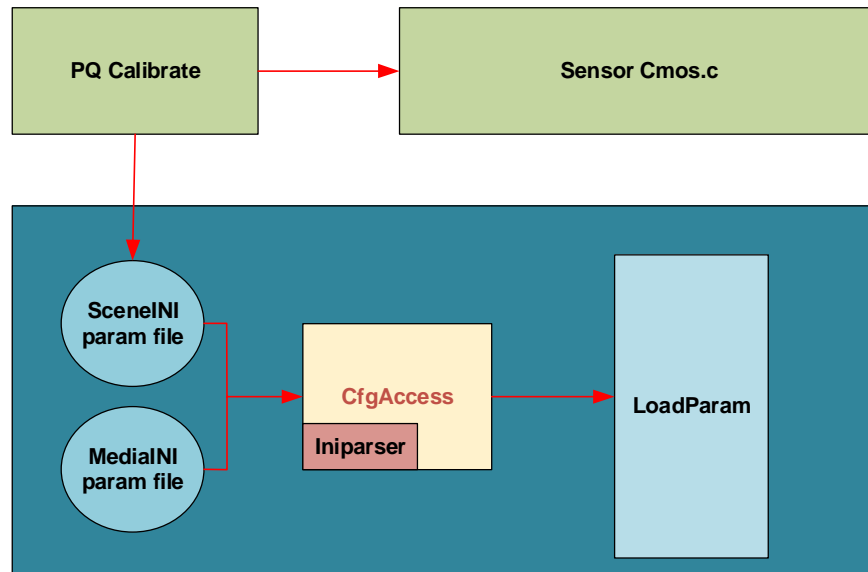
1.2 Scene-auto Sample Module Overview

The scene-auto module can be divided into the following submodules by function:

- Import of the scene-auto parameters
- Main scene-auto framework
- Scene-auto adjustment module

1.2.1 Import of the Scene-auto Parameters

Figure 1-3 Obtaining scene-auto sample parameters



The scene-auto debugging parameters are obtained by PQ offline calibration. During the running of the scene-auto module, ensure that all the scene-auto debugging parameters are available.

The parameters for offline PQ calibration can be stored in the .ini configuration file or the source code. When the parameters are stored in the .ini configuration file, you can use the .ini parser to obtain the parameters. In this way, you do not need to compile the source code during parameter debugging. When the parameters are stored in the source code, the obtained parameters do not need to be parsed. However, the parameters need to be recompiled during parameter debugging.



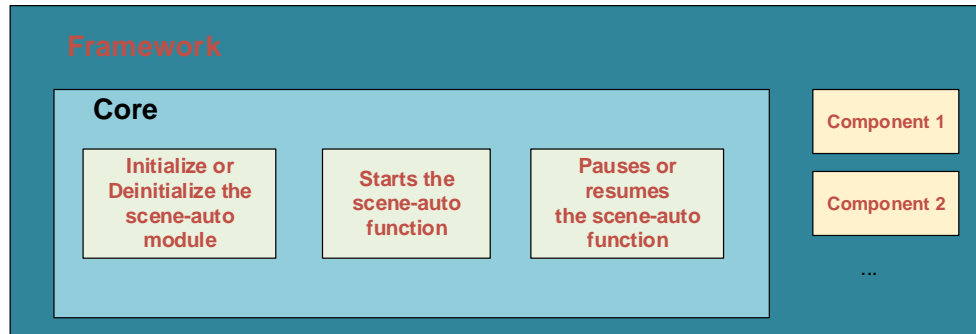
NOTE

- The scene-auto debugging parameters can also be obtained by production calibration during factory test, for example, the mesh-shading table and the table of static defect pixels.
- Such debugging parameters can only be stored in the .ini configuration file. Therefore, you do not need to recompile the source code during parameter debugging as well.

1.2.2 Main Scene-auto Framework

The main scene-auto framework is a scene-auto submodule that provides the function interfaces for the product. They can be used to initialize and deinitialize the scene-auto module, update the scene-auto parameters according to the current media channel, and enable the scene-auto parameter adjustment function.

This submodule applies to multiple product forms, including but not limited to the IP camera (IPC) and mobile cam. The basic functions mentioned above are provided for different product forms.

Figure 1-4 Scene-auto framework code

NOTE

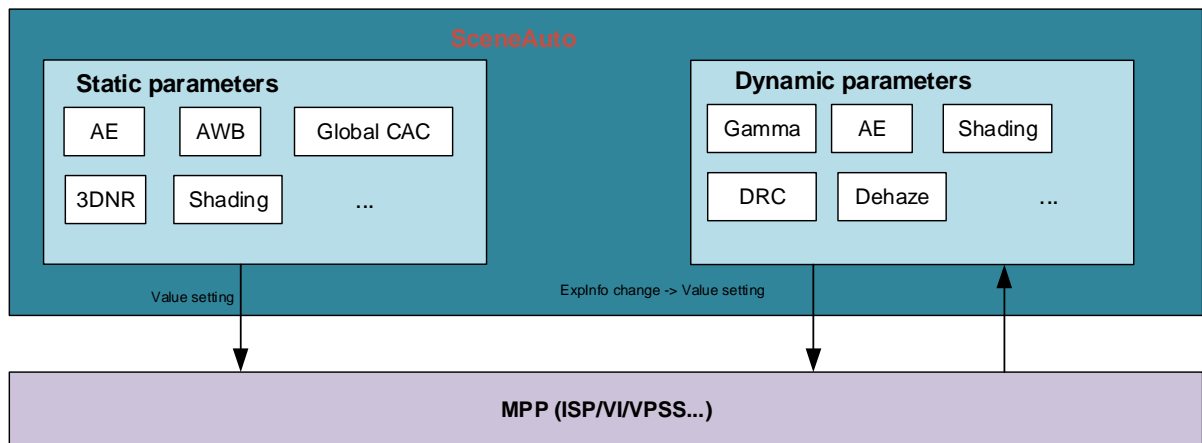
Component 1, 2 ... are reserved and will include new specifications in future.

1.2.3 Scene-auto Adjustment Module

Scene-auto parameters can be divided into static and dynamic parameters.

- Static parameters need to be configured only once before parameters take effect.
- Dynamic parameter settings vary with the scene. Static and dynamic parameters can be further divided based on the ISP modules.

For sensors of different product forms or the same product form, the scene-auto root parameters to be debugged are different. Therefore, the value assignment for the scene-auto debugging parameters is extracted as a separate module. In this way, for scene-auto module as a whole, only this submodule is subject to change, greatly reducing the complexity of code maintenance.

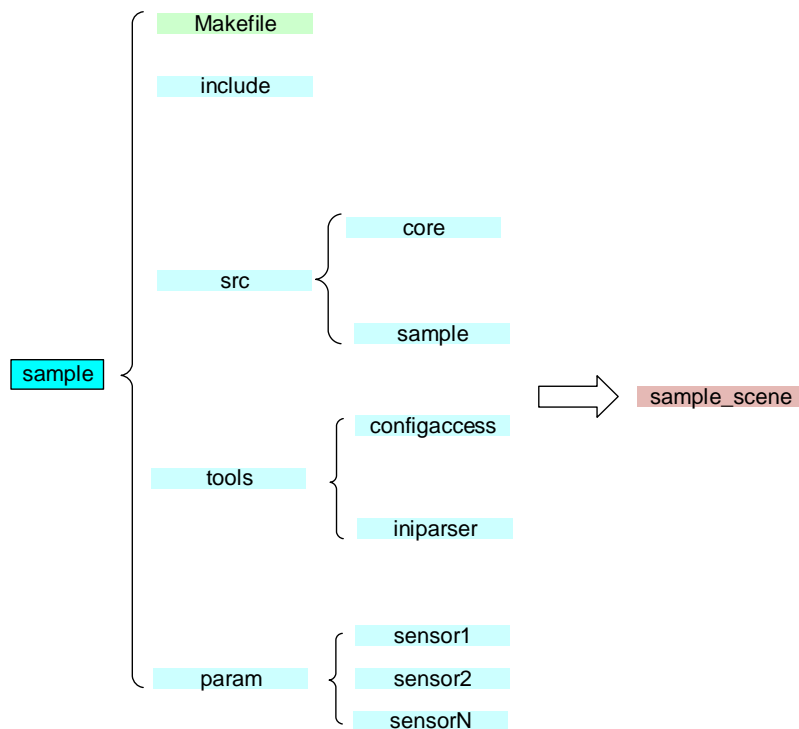
Figure 1-5 Scene-auto adjustment module


1.3 File Structure

Figure 1-6 shows the file structure of the scene-auto module. Files are organized around the tools (the .ini parsing component **configaccess** and the .ini parser **iniparser**), parameters, sources, and library files for different sensors, and the common code of the entire module.

- The **tools** folder contains the **cfgaccess** tool, which is encapsulated based on **iniparser** and is used to parse the .ini file.
- The **src** folder contains the core (scene-auto framework code), and sample (support code of the scene-auto sample), and multiple sensor sub-directories (in which the .ini parameters and the scene-auto adjustment code generated by using the ini parameters are placed).
- The **param** folder stores .ini files for sensor parameters.

Figure 1-6 File organization of the scene-auto module

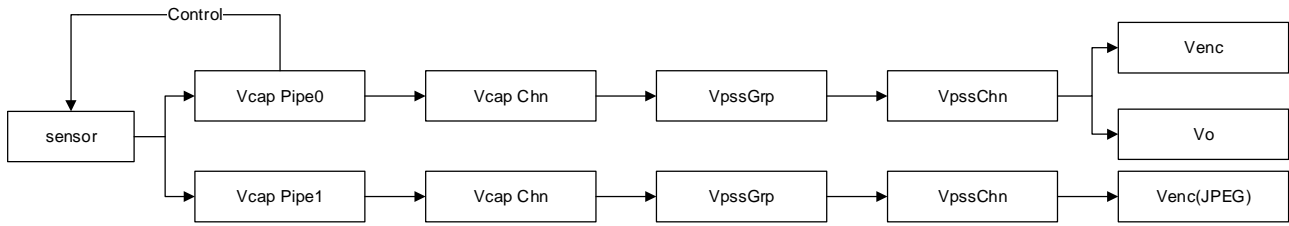


1.4 Important Concepts

The following takes the dual-pipe snapshot mode (in the case of 8M snapshot and the front-end sensor at 4K@30 fps) as an example to describe how the scene-auto module adjusts image parameters and the concepts of API data structures.

Figure 1-7 shows the pipeline for preview and snapshot with a single sensor.

Figure 1-7 Pipeline for preview and snapshot with a single sensor



As shown in [Figure 1-7](#), the data of both pipes in the current media pipeline comes from the same sensor, and the ISP in pipe 0 controls the sensor. Therefore, **MainPipeHdl** of pipe 0 and **MainPipeHdl** of pipe 1 are both 0.

- **MainPipeHdl** indicates that the number of pipes whose ISP can control the sensor is the same as the number of sensors. When the data of both pipes comes from the same sensor, the **MainPipeHdl** values of both pipes are the same.
- For the pipeline for preview and snapshot with a single sensor, to adjust the image quality, you need to configure the ISP in the sub-pipeline, three-dimensional noise reduction (3DNR) of the video processing subsystem (VPSS), and encoding attributes of the VENC.
- A sub-pipeline goes through the video capture (VCAP) pipe, the VPSS bound to the VCAP pipe, and the subsequent VENC and video output (VO) modules. Therefore, there are two sub-pipelines in [Figure 1-7](#).

The scene-auto module divides the current media pipeline into one or more sub-pipelines. Each sub-pipeline is configured with a group of scene-auto parameters. Each group of scene-auto parameters is maintained by an image parameter adjustment file. For details, see [4 "Data Structure."](#) The scene-auto parameters that need to be configured for the current sub-pipeline are specified by **u8PipeParamIndex**. For details about **u8PipeParamIndex**, see [HI_SCENE_PIPE_ATTR_S](#).



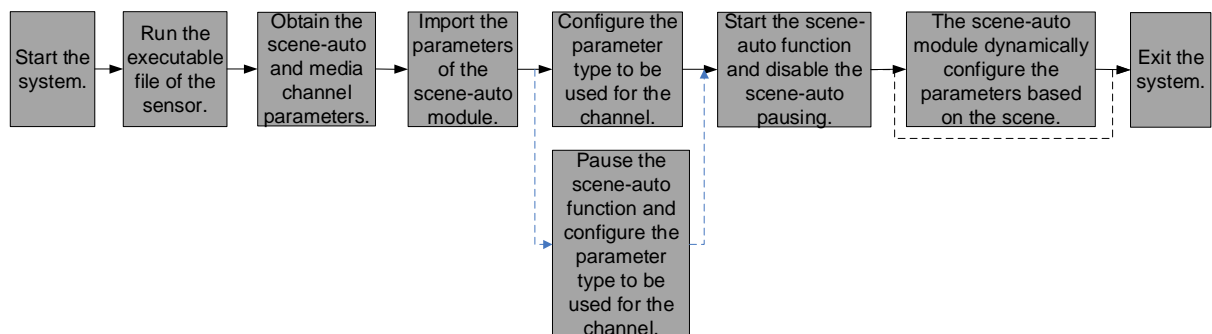
2 Development Guide

2.1 Usage Procedure

2.1.1 Overview

The following describes the APIs and parameters of the scene-auto module. For details, see the scene-auto SDK sample. [Figure 2-1](#) shows the scene-auto process.

Figure 2-1 Process of using the scene-auto module



2.1.2 Running the Sensorini File

Scene

In the SDK sample, each sensor has its own folder and all the folders are independent of each other. Parameters are maintained by the sensor.

Workflow

The execution workflow is as follows:

- Step 1** In the **Sceneauto** directory, run **Makefile** to generate the corresponding **sample_scene**.
 - Step 2** Run **sample_scene** to load the .ini file of the corresponding sensor in the **param** directory.
- End



Precautions

- For details about parameter settings in the **param** file of the **Sceneauto** directory, see section 2.2 "Running the Sample with a New Sensor."
- The source code in the **source** file of the **Src/core** directory is used to assign values to the scene-auto parameters.

2.1.3 Obtaining and Importing Scene-auto and Media Channel Parameters

Scene

The parameters required for enabling the scene adaptation function are divided into scene-auto debugging parameters and media channel parameters. In the sensor directory, there are multiple .ini files of scene-auto debugging parameters. For example, if the scene-auto function is enabled, a group of scene-auto debugging parameters should be assigned to each running pipe on the media channels. The parameters assigned to each pipe can be the same or different.

In the sample, the scene-auto debugging parameters and media channel parameters are stored in the .ini configuration files. You need to parse the .ini file, write the parameters to the structure in the specified memory, and import the data to the scene-auto module.

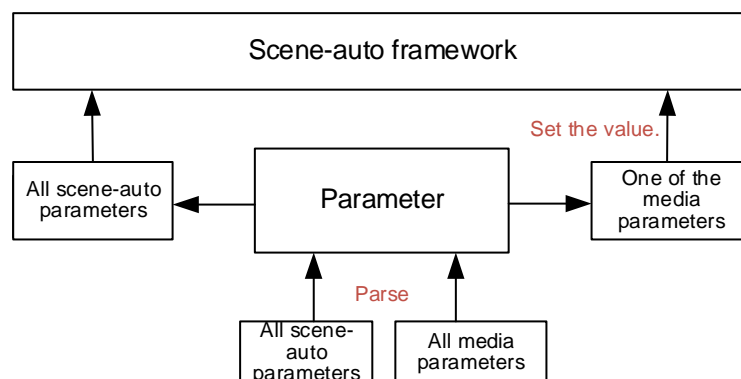
Workflow

Perform the following steps to obtain and import the parameters required by the module:

- Step 1** **Configaccess** parses the **config_cfgaccess_hd.ini** file in the sensor directory.
- Step 2** **Configaccess** parses the scene-auto debugging parameters and media channel parameters, and stores the two types of parameters in separate arrays.
- Step 3** The sample calls the scene-auto interfaces to import the scene-auto debugging parameter array into the scene-auto module and applies a set of parameter configuration that matches the current media channel to the module.

----End

Figure 2-2 Process of using the scene-auto module





Precautions

- The media channel parameters are described in detail in the data structure section. The media channel parameters are mainly used to enable or disable the current media channel and indicate the set of scene-auto debugging parameters that should be configured on each enabled media pipe.
- The set of scene-auto debugging parameters is an array. The scene-auto debugging parameters to be configured for each pipe in the media channel parameters are the subscripts of the array, which need to be specified in advance.
- The positions of the scene-auto debugging parameters in the array are specified in **config_cfgaccess_hd.ini**.

2.1.4 Enabling the Scene-auto Function

Scene

To enable the scene-auto function, perform the following two steps. Configure the static parameters on each pipe. The static parameters need to be set once only. Then, configure dynamic parameters on each pipe. The dynamic parameters dynamically change with the scene.



NOTE

In the current version, the dynamic parameters change according to the exposure, ISO, or exposure time ratio in the current scene.

Workflow

The procedure for enabling scene adaptation is as follows:

- Step 1** Call `HI_SCENE_SetSceneMode()` to configure the media channel parameters.
 - Step 2** Enable scene adaptation, and dynamically adjust certain parameters according to the current scene.
- End

2.1.5 Switching Scene-auto Configuration

Scene

In the scene adaptation process, typical behaviors such as media channel change is accompanied. When the media channel changes, for example, the linear mode is switched to the WDR mode or resolution change in motion DV mode, the scene-auto parameters to be configured on each channel pipe need to change accordingly. In this case, the scene-auto configuration needs to be switched.

Workflow

The procedure for switching scene-auto configuration is as follows:

- Step 1** Pause adaptation by calling `HI_SCENE_Pause(HI_TRUE)`.
- Step 2** Switch the media channel.
- Step 3** Obtain the scene-auto configuration parameters for this media channel.



Step 4 Enable the scene-auto function by calling `HI_SCENE_SetSceneMode()`.

Step 5 Resume the scene-auto function by calling `HI_SCENE_Pause(HI_FALSE)`.

----End

Precautions

- When the media channel is disabled, you should call scene-auto interface `HI_SCENE_Pause(HI_TRUE)` to prevent the ISP interfaces from being called.
- When the media channel is enabled, you should call scene-auto interface `HI_SCENE_Pause(HI_FALSE)` to enable the calling of the ISP interfaces.

2.1.6 .ini Parser

In the current scene-auto sample, the .ini file parsing depends on the open source .ini file parsing C library **iniparser**, and **cfgaccess**, which is encapsulated based on **iniparser** by HiSilicon.

iniparser parses the character strings in the .ini configuration file into specific data such as the bool values, int values, and long values. **cfgaccess** can be used to manage the .ini configuration files in the primary .ini configuration file. In the scene-auto sample, the primary .ini configuration file is **config_cfgaccess_hd.ini** in the **param** directory of each sensor, as shown in [Figure 2-3](#).

Figure 2-3 .ini configuration management files required by **cfgaccess**

```
[module]
module_num      = "3"
module1         = "video_mode_0"
module2         = "scene_mode_0"
module3         = "scene_mode_1"

[video_mode_0]
path            = "./imx117/param/"
cfg_filename    = "config_videomode_0.ini"

[scene_mode_0]
path            = "./imx117/param/"
cfg_filename    = "config_product_scene_1080p30_linear.ini"

[scene_mode_1]
path            = "./imx117/param/"
cfg_filename    = "config_product_scene_1080p30_3t1_wdr.ini"
```

There is only one .ini configuration file for the media parameters corresponding to **video_mode_0**. The configuration of the media parameters is stored in this file. There are multiple scene-auto configuration parameters corresponding to **scene_mode_n**, and each .ini configuration file is dedicated to a group of parameters.

Through this file, the .ini parser can parse multiple .ini configuration files at a time and store the parameters generated by the .ini configuration file in order.

Also, you can use **cfgaccess** to check whether the .ini configuration file contains the required items. For details, see the sample code.



2.1.7 Scene-auto Configuration Parameters

Scene-auto configuration parameters are classified into dynamic and static configuration parameters. They can be further divided into subclasses by module, such as static AE parameters, dynamic 3DNR parameters, and dynamic AE parameters. Figure 2-4 lists the .ini configuration file of each type of dynamic AE parameters.

Figure 2-4 .ini configuration file for the dynamic AE parameters

```

L
//dynamic_AE_parameter
[dynamic_ae]
aeExpCount          = "5"
aeExpLtoHThresh     = "10000, 40000, 300000, 1000000, 10000000"
aeExpHtoLThresh     = "5000, 7000, 30000, 200000, 800000"
AutoCompensation    = "30, 28, 24, 24, 24"
AutoHistOffset      = "20, 20, 20, 20, 20"

```

aeExpCount indicates the array size, **ExpLtoH** and **ExpHtoL** indicate the arrays of exposure values, and **Compensation** and **HistOffset** correspond to parameters in the exposure values configured in the ISP.

You can learn the module functions from the module names. For example, the **dynamic_ae** module configures the dynamic AE parameters. The **ExpThresh** module indicates that the dynamic AE parameters are determined by the exposure of the current environment. The dynamic AE adjustment values are the ISP parameters corresponding to **Compensation** and **HistOffset**.



NOTE

- An .ini scene-auto configuration file can be filled in the scene-auto parameter array through parameter parsing. Each pipe can be configured with one item of the scene-auto parameter array.
- Ensure that the number of .ini scene-auto configuration files of a sensor in the sample is within the threshold to prevent out-of-bounds access of arrays.

2.1.8 Parameters for Configuring the Media Channel

The parameters for configuring the media channel need to be sent to the scene-auto framework code interface as an input argument. Figure 2-5 shows a set of typical values of a group of parameters for configuring the media channel.

The parameters for configuring the media channel can tell whether the linear or WDR mode is used, whether all pipes are enabled, and which group of scene-auto parameters should be configured on the pipe (specified by **PipeParamIndex**).



NOTE

- **PipeParamIndex** in the ini configuration file refers to the subscript of the scene-auto configuration parameter array.
- The arrangement of the array of the scene-auto configuration parameters is specified in the .hd file as shown in Figure 2-3. **scene_mode_0** corresponds to array subscript 0 while **scene_mode_1** corresponds to array subscript 1.

Multiple groups of media channels are configured in the same .ini configuration file. When the scene-auto function interface is called, only one set of configuration is required. During the adaptation, the configuration plan can be dynamically switched.

**Figure 2-5** .ini configuration file of the media channel parameters

```

[pipe_comm_0]
SCENE_MODE           = "0"           ;;0- linear, 1-wdr, 2-hdr

[pipe_0_0]
Enable               = "1"           ;;0 - disable, 1 - able
VcapPipeHdl          = "0"           ;;Pipe Handle, must equals to pipe_X;
MainPipeHdl          = "0"           ;;MainIsp Handle
PipeChnHdl           = "0"           ;;Pipe Chn
VpssHdl              = "0"           ;;VpssGrpHdl
VportHdl             = "0"           ;;VpssChnHdl
VencHdl              = "0"           ;; Not Used Currently
PipeParamIndex       = "0"           ;; Pipe Use Which Param
PipeType             = "1"           ;;0 - snap, 1-video

[pipe_0_1]
Enable               = "0"           ;;0 - disable, 1 - able
VcapPipeHdl          = "1"           ;;Pipe Handle, must equals to pipe_X;
MainPipeHdl          = "0"           ;;MainIsp Handle
PipeChnHdl           = "0"           ;;Pipe Chn
VpssHdl              = "0"           ;;VpssGrpHdl
VportHdl             = "0"           ;;VpssChnHdl
VencHdl              = "0"           ;; Not Used Currently
PipeParamIndex       = "0"           ;; Pipe Use Which Param
PipeType             = "1"           ;;0 - snap, 1-video

[pipe_0_2]
Enable               = "0"           ;;0 - disable, 1 - able
VcapPipeHdl          = "2"           ;;Pipe Handle, must equals to pipe_X;
MainPipeHdl          = "0"           ;;MainIsp Handle
PipeChnHdl           = "0"           ;;Pipe Chn
VpssHdl              = "0"           ;;VpssGrpHdl
VportHdl             = "0"           ;;VpssChnHdl
VencHdl              = "0"           ;; Not Used Currently
PipeParamIndex       = "0"           ;; Pipe Use Which Param
PipeType             = "1"           ;;0 - snap, 1-video

```

2.2 Running the Sample with a New Sensor

Scene

The scene-auto module is compatible with multiple types of sensors. The sample uses several typical sensors. When a sensor needs to be added, the following adaptation is required.

Workflow

The process of adapting a sensor is as follows:

- Step 1** Create a folder named **sensorXXXX** in the **param** directory in the home directory of the sample.
- Step 2** Place the to-be-adjusted parameters of the sensor in the **sensorXXXX** folder.







You can copy the files in the **param** folder of another sensor to the **sensorXXXX** folder and add or delete parameter types.

Step 3 Modify the path names in the **config_cfgaccess_hd.ini** file in the **param** folder.

Step 4 Modify the .ini parameters in the **param** folder.

Add the scene-auto configuration parameters of the sensor. For details, see **config_product_scene_1080p30_3t1_wdr** in [Figure 2-6](#). You are advised to name the parameters by resolution frame rate and linear WDR mode for convenience. For details about how to add or delete parameters, see [section 2.3 "Running the Sample with Parameters Added or Deleted."](#)

Figure 2-6 Contents in the param folder of the sensor

 config_cfgaccess_hd	2018/2/27 20:09	Configuration Se...	1 KB
 config_product_scene_1080p30_3t1_wdr	2018/2/26 16:52	Configuration Se...	80 KB
 config_product_scene_1080p30_linear	2018/2/26 16:52	Configuration Se...	116 KB
 config_videomode_0	2018/2/24 14:41	Configuration Se...	10 KB

Modify the content in **config_videomode_0** based on the configuration of the media channel used by the sensor.

Modify the content in the **hd.ini** file.

Step 5 Run the **make clean; make** command in the home directory to generate the executable file for the scene-auto sample.

----End

2.3 Running the Sample with Parameters Added or Deleted

Scene

During PQ debugging, parameters are frequently modified, added or deleted to achieve the optimal image effect. The following describes how the scene-auto module works when parameters are added or deleted.

The following procedure shows how to add a parameter to the .ini configuration items.

Workflow

The procedure for adding a parameter is as follows:

Step 1 Determine the parameter to be added for the sensor and the required ISP interface data structure.

Step 2 Add the parameter to the .ini file in the **param** folder of the sensor and rename the parameter as required. For example, add a minimum system gain to **static_ae**, as shown in [Figure 2-7](#).

**Figure 2-7** Adding a parameter to the scene-auto configuration items

```

//////////////////////////////////// static_AE_parameter //////////////////////////////////////
[static_ae]
AERouteExValid      = "1"
AERunInterval       = "1"
AutoSpeed            = "64"
AutoTolerance        = "2"
AutoBlackDelayFrame  = "8"
AutoWhiteDelayFrame  = "0"
AutoSysGainMax       = "65536"

//////////////////////////////////// static_AE_parameter //////////////////////////////////////
[static_ae]
AERouteExValid      = "1"
AERunInterval       = "1"
AutoSpeed            = "64"
AutoTolerance        = "2"
AutoBlackDelayFrame  = "8"
AutoWhiteDelayFrame  = "0"
AutoSysGainMax       = "65536"
AutoSysGainMin       =

```

AutoSysGainMin in the .ini configuration file requires **stAuto.stSysGainRange.u32Min** in the **ISP_EXPOSURE_ATTR_S** structure of the SDK interface. You need to have a comprehensive understanding of the mapping relationship.

Note that this parameter must be added to all the scene-auto .ini configuration files of the sensor.

- Step 3** Add a parameter to the data structure **HI_SCENE_STATIC_AE_S** in the **hi_scene_setparam.h** file of the **include** folder under the home directory of the sample. This added parameter corresponds to **AutoSysGainMin** in the .ini configuration file. The parameter name is user-defined, as shown in [Figure 2-8](#).

Figure 2-8 Adding a parameter to the scene-auto header file for value assignment

```

typedef struct hiSCENE_STATIC_AE_S typedef struct hiSCENE_STATIC_AE_S
{
    HI_BOOL bAERouteExValid;
    HI_U8 u8AERunInterval;
    HI_U32 u32AutoSysGainMax;
    HI_U8 u8AutoSpeed;
    HI_U8 u8AutoTolerance;
    HI_U16 u16AutoBlackDelayFrame;
    HI_U16 u16AutoWhiteDelayFrame;
} HI_SCENE_STATIC_AE_S;

{
    HI_BOOL bAERouteExValid;
    HI_U8 u8AERunInterval;
    HI_U32 u32AutoSysGainMax;
    HI_U32 u32AutoSysGainMin;
    HI_U8 u8AutoSpeed;
    HI_U8 u8AutoTolerance;
    HI_U16 u16AutoBlackDelayFrame;
    HI_U16 u16AutoWhiteDelayFrame;
} HI_SCENE_STATIC_AE_S;

```

- Step 4** Add the statement of value assignment for the parameter shown in [Figure 2-9](#) to **hi_scene_setparam.c** in the **src/core** folder of the sensor.

**Figure 2-9** Adding a parameter to the scene-auto source file for value assignment

```
HI_S32 HI_SCENE_SetStaticAE(VI_PIPE ViPipe, HI_U8 u8Index)
{
    if(HI_TRUE != g_astScenePipeParam[u8Index].stModuleState.bStaticAE)
    {
        return HI_SUCCESS;
    }

    HI_S32 i = 0;
    HI_S32 s32Ret = HI_SUCCESS;

    ISP_PUB_ATTR_S stPubAttr;
    ISP_EXPOSURE_ATTR_S stExposureAttr;
    ISP_AE_ROUTE_EX_S stAeRouteEx;

    s32Ret = HI_MPI_ISP_GetAERouteAttrEx(ViPipe, &stAeRouteEx);
    CHECK_SCENE_RET(s32Ret);

    stAeRouteEx.u32TotalNum = g_astScenePipeParam[u8Index].stStaticAeRouteEx.u32TotalNum;
    for (i = 0; i < stAeRouteEx.u32TotalNum; i++)
    {
        stAeRouteEx.astRouteExNode[i].u32IntTime = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IntTime[i];
        stAeRouteEx.astRouteExNode[i].u32Again = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32Again[i];
        stAeRouteEx.astRouteExNode[i].u32Dgain = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32Dgain[i];
        stAeRouteEx.astRouteExNode[i].u32IspDgain = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IspDgain[i];
    }

    s32Ret = HI_MPI_ISP_SetAERouteAttrEx(ViPipe, &stAeRouteEx);
    CHECK_SCENE_RET(s32Ret);

    s32Ret = HI_MPI_ISP_GetExposureAttr(ViPipe, &stExposureAttr);
    CHECK_SCENE_RET(s32Ret);

    s32Ret = HI_MPI_ISP_GetPubAttr(ViPipe, &stPubAttr);
    CHECK_SCENE_RET(s32Ret);

    i = g_astScenePipeParam[u8Index].stStaticAeRouteEx.u32TotalNum - 1;
    stExposureAttr.stAuto.stExpTimeRange.u32Max = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IntTime[i];

    if (WDR_MODE_NONE != stPubAttr.enWDRMode)
    {
        stExposureAttr.stAuto.stDgainRange.u32Max = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32Dgain[i];
        stExposureAttr.stAuto.stISPDgainRange.u32Max = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IspDgain[i];
    }

    stExposureAttr.bAERouteExValid = g_astScenePipeParam[u8Index].stStaticAe.bAERouteExValid;
    stExposureAttr.u8AERunInterval = g_astScenePipeParam[u8Index].stStaticAe.u8AERunInterval;
    stExposureAttr.stAuto.stSysGainRange.u32Max = g_astScenePipeParam[u8Index].stStaticAe.u32AutoSysGainMax;

HI_S32 HI_SCENE_SetStaticAE(VI_PIPE ViPipe, HI_U8 u8Index)
{
    if(HI_TRUE != g_astScenePipeParam[u8Index].stModuleState.bStaticAE)
    {
        return HI_SUCCESS;
    }

    HI_S32 i = 0;
    HI_S32 s32Ret = HI_SUCCESS;

    ISP_PUB_ATTR_S stPubAttr;
    ISP_EXPOSURE_ATTR_S stExposureAttr;
    ISP_AE_ROUTE_EX_S stAeRouteEx;

    s32Ret = HI_MPI_ISP_GetAERouteAttrEx(ViPipe, &stAeRouteEx);
    CHECK_SCENE_RET(s32Ret);

    stAeRouteEx.u32TotalNum = g_astScenePipeParam[u8Index].stStaticAeRouteEx.u32TotalNum;
    for (i = 0; i < stAeRouteEx.u32TotalNum; i++)
    {
        stAeRouteEx.astRouteExNode[i].u32IntTime = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IntTime[i];
        stAeRouteEx.astRouteExNode[i].u32Again = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32Again[i];
        stAeRouteEx.astRouteExNode[i].u32Dgain = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32Dgain[i];
        stAeRouteEx.astRouteExNode[i].u32IspDgain = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IspDgain[i];
    }

    s32Ret = HI_MPI_ISP_SetAERouteAttrEx(ViPipe, &stAeRouteEx);
    CHECK_SCENE_RET(s32Ret);

    s32Ret = HI_MPI_ISP_GetExposureAttr(ViPipe, &stExposureAttr);
    CHECK_SCENE_RET(s32Ret);

    s32Ret = HI_MPI_ISP_GetPubAttr(ViPipe, &stPubAttr);
    CHECK_SCENE_RET(s32Ret);

    i = g_astScenePipeParam[u8Index].stStaticAeRouteEx.u32TotalNum - 1;
    stExposureAttr.stAuto.stExpTimeRange.u32Max = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IntTime[i];

    if (WDR_MODE_NONE != stPubAttr.enWDRMode)
    {
        stExposureAttr.stAuto.stDgainRange.u32Max = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32Dgain[i];
        stExposureAttr.stAuto.stISPDgainRange.u32Max = g_astScenePipeParam[u8Index].stStaticAeRouteEx.au32IspDgain[i];
    }

    stExposureAttr.bAERouteExValid = g_astScenePipeParam[u8Index].stStaticAe.bAERouteExValid;
    stExposureAttr.u8AERunInterval = g_astScenePipeParam[u8Index].stStaticAe.u8AERunInterval;
    stExposureAttr.stAuto.stSysGainRange.u32Max = g_astScenePipeParam[u8Index].stStaticAe.u32AutoSysGainMax;
    stExposureAttr.stAuto.stSysGainRange.u32Min = g_astScenePipeParam[u8Index].stStaticAe.u32AutoSysGainMin;
```




Step 5 Add the statement shown in [Figure 2-10](#) to the SCENE_LoadStaticAE function in `scene_loadparam.c` in the scene-auto home directory.

Figure 2-10 Adding a parameter to the scene-auto source file for value obtaining

```
snprintf(aszIniNodeName, SCENE_INIPARAM_NODE_NAME_LEN, "static_ae:AutoSysGainMin"); /*AutoSysGainMin*/
s32Ret = HI_CONFACCESS_GetString(SCENE_INIPARAM, pszIniModule, aszIniNodeName, NULL, &pszString);
SCENE_INIPARAM_CHECK_LOAD_RESULT(s32Ret, aszIniNodeName);
if (HI_NULL != pszString)
{
    free(pszString);
    pszString = HI_NULL;
    s32Ret = HI_CONFACCESS_GetInt(SCENE_INIPARAM, pszIniModule, "static_ae:AutoSysGainMin", 0, &s32Value);
    if (HI_SUCCESS != s32Ret)
    {
        MLOGE("load static_ae:AutoSysGainMin failed\n");
        return HI_FAILURE;
    }
    pstStaticAe->u32AutoSysGainMax = (HI_U32)s32Value;
}
```

Step 6 To simplify parameter debugging, you only need to perform the reverse operation of adding a sensor. However, the code in `scene_loadparam.c` does not need to be modified because this is a common code shared by all sensors. Addition to the code does not influence the code of other sensors. However, decrement from the code influences the code implementation of other sensors.

----End

2.4 Typical Pipeline Configuration Solutions

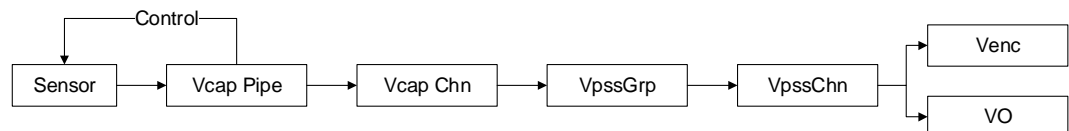
To adaptively adjust the image quality, you need to call `HI_SCENE_SetSceneMode`. The input parameters of this API vary according to the type of the media pipeline. Unless otherwise specified, the ISP is in linear mode. The following describes how to configure `HI_SCENE_MODE_S` based on the four typical types of media pipelines.

2.4.1 Preview and Video Recording with a Single Sensor

Assume that the pipe is pipe 0 and the working mode is 4K@30 fps recording. One pipe is enabled in the media pipeline. In addition, the ISP in this pipe controls the sensor. Therefore, the `MainPipeHdl` count is 1 and is the same as the count of the enabled handle `VcapPipeHdl`.

[Figure 2-11](#) shows the pipeline for preview and video recording with a single sensor.

Figure 2-11 Pipeline for preview and video recording with a single sensor



For the media pipeline shown in [Figure 2-11](#), the parameter settings of `HI_SCENE_MODE_S` are required, as shown in [Table 2-1](#).

**Table 2-1** Parameter configuration A of HI_SCENE_MODE_S

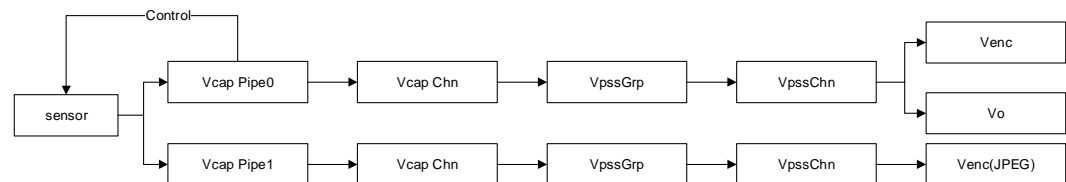
Parameter Type	astPipeAttr[0]	astPipeAttr[1]
bEnable	1	0
VcapPipeHdl	0	default
MainPipeHdl	0	default
u8PipeParamIndex	0	default
enPipeType	VIDEO	default

Other values are configured based on the actual status of the pipeline.

2.4.2 Preview and Snapshot with a Single Sensor

Assume that the pipe in the preview sub-pipeline is pipe 0 and the pipe in the snapshot sub-pipeline is pipe 1. In this case, the working mode is 8M snapshot. Two pipes are enabled in the media pipeline. The data of both pipes comes from the same sensor. Therefore, the **MainPipeHdl** count is 1 as well. The pipe whose ISP controls the sensor is in the preview pipeline.

For the media pipeline shown in [Figure 2-12](#), the parameter settings of **HI_SCENE_MODE_S** are required, as shown in [Table 2-2](#).

Figure 2-12 Pipeline for preview and snapshot with a single sensor**Table 2-2** Parameter configuration B of HI_SCENE_MODE_S

Parameter Type	astPipeAttr[0]	astPipeAttr[1]
bEnable	1	1
VcapPipeHdl	0	1
MainPipeHdl	0	0
u8PipeParamIndex	1	2
enPipeType	VIDEO	SNAP

Other values are configured based on the actual status of the pipeline.

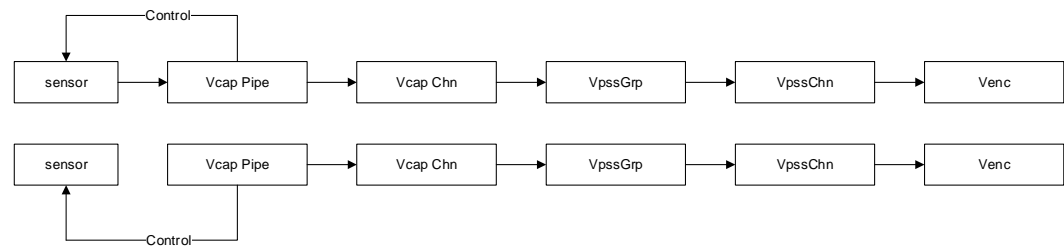


2.4.3 Preview and Video Recording with Multiple Sensors

Assume that the pipes in the preview sub-pipelines are pipes 0 and 1. In addition, the working mode is 4K@30 fps with front and rear preview and video recording pipelines. Two pipes are enabled in the media pipeline. The data of two pipes comes from two different sensors. Therefore, the **MainPipeHdl** count is 2 as well. The pipe whose ISP controls the sensor is in the preview pipeline.

Figure 2-13 shows the pipeline for preview and video recording with multiple sensors.

Figure 2-13 Pipeline for preview and video recording with multiple sensors



For the media pipeline shown in Figure 2-13, the parameter settings of **HI_SCENE_MODE_S** are required, as shown in Table 2-3.

Table 2-3 Parameter configuration C of HI_SCENE_MODE_S

Parameter Type	astPipeAttr[0]	astPipeAttr[1]
bEnable	1	1
VcapPipeHdl	0	1
MainPipeHdl	0	1
u8PipeParamIndex	0	0
enPipeType	VIDEO	VIDEO

Other values are configured based on the actual status of the pipeline.

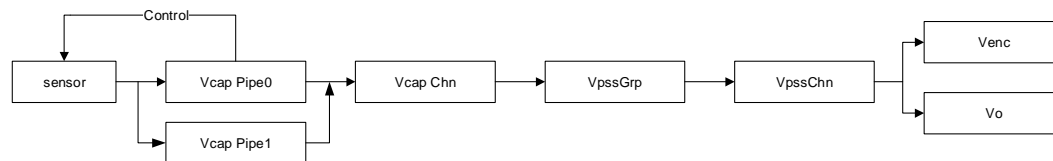
2.4.4 WDR Preview and Video Recording with a Single Sensor

The 2-to-1 WDR mode requires two pipes, but only one ISP enabled. Therefore, there is only one pipeline/sub-pipeline for scene-auto.

Assume that the pipe is pipe 0 and the working mode is 4K@30 fps recording.

Two pipes are enabled in the media pipeline. The data of both pipes comes from the same sensor. Therefore, the **MainPipeHdl** count is 1 as well. The pipe whose ISP controls the sensor is VCAP pipe 0

Figure 2-14 shows the pipeline for preview and video recording in 2-to-1 WDR mode with a single sensor.

**Figure 2-14** Pipeline for preview and video recording in 2-to-1 WDR mode with a single sensor

For the media pipeline shown in [Figure 2-14](#), the parameter settings of [HI_SCENE_MODE_S](#) are required, as shown in [Table 2-4](#).

Table 2-4 Parameter configuration D of HI_SCENE_MODE_S

Parameter Type	astPipeAttr[0]	astPipeAttr[1]
bEnable	1	0
VcapPipeHdl	0	default
MainPipeHdl	0	default
u8PipeParamIndex	1	default
enPipeType	VIDEO	default



3 API Reference

The scene-auto module provides the following APIs:

- [HI_SCENE_Init](#): Initializes the scene-auto module.
- [HI_SCENE_SetSceneMode](#): Configures scene-auto media parameters and enables the scene-auto function.
- [HI_SCENE_Pause](#): Pauses or resumes the scene-auto function.
- [HI_SCENE_GetSceneMode](#): Obtains the scene-auto media parameters.
- [HI_SCENE_Deinit](#): Deinitializes the scene-auto module.

HI_SCENE_Init

[Description]

Initializes the scene-auto module and imports scene-auto configuration parameters.

[Syntax]

```
HI_S32 HI_SCENE_Init(const HI\_SCENE\_PARAM\_S* pstSceneParam)
```

[Parameter]

Parameter	Description	Input/Output
pstSceneParam	Set of the scene-auto configuration parameters	Input

[Return Value]

Return Value	Description
0	Success
Other value	Failure. For details, see section 5 " Error Code ."

[Requirement]

- Header file: **hi_scene.h**, hi_scene_autogenerate.h
- Library file: null



[Note]

- **pstSceneParam** cannot be blank.
- Up to ten scene-auto parameters can be configured.

[Example]

scene_sample.c

HI_SCENE_SetSceneMode

[Description]

Configures scene-auto media parameters and enables the scene-auto function.

[Syntax]

```
HI_S32 HI_SCENE_SetSceneMode (const HI_SCENE_MODE_S* pstSceneMode)
```

[Parameter]

Parameter	Description	Input/Output
pstSceneMode	Scene-auto media configuration for the current media channel	Input

[Return Value]

Return Value	Description
0	Success
Other value	Failure. For details, see section 5 " Error Code ."

[Requirement]

- Header file: **hi_scene.h**
- Library file: null

[Note]

- **pstSceneMode** cannot be null.
- This API cannot be called in multi-thread mode.
- This API must be called after the scene-auto module is initialized.
- The enabling status of the current media channel must be consistent with that of the scene-auto media parameters.

[Example]

scene_sample.c



HI_SCENE_Pause

[Description]

Pauses or resumes the scene-auto function.

[Syntax]

```
HI_S32 HI_SCENE_Pause (HI_BOOL bEnable)
```

[Parameter]

Parameter	Description	Input/Output
bEnable	True: The scene-auto function is paused. False: The scene-auto function is resumed.	Input

[Return Value]

Return Value	Description
0	Success
Other value	Failure. For details, see section 5 " Error Code ."

[Requirement]

- Header file: **hi_scene.h**
- Library file: null

[Note]

- Before the external media channel is destroyed, the scene-auto function must be paused. The ISP attribute is continuously configured by the scene-auto function. If the ISP is stopped due to channel destruction, an error is reported when the ISP is configured.
- After the external media channel is enabled, the scene-auto function needs to be resumed to use the scene-auto function.
- This API must be called after the scene-auto module is initialized.

[Example]

scene_sample.c

HI_SCENE_GetSceneMode

[Description]

Obtains the scene-auto media parameters.

[Syntax]

```
HI_S32 HI_SCENE_GetSceneMode (HI_SCENE_MODE_S* pstSceneMode)
```

[Parameter]



Parameter	Description	Input/Output
pstSceneMode	Current scene-auto media configuration	Output

[Return Value]

Return Value	Description
0	Success
Other value	Failure. For details, see section 5 " Error Code ."

[Requirement]

- Header file: **hi_scene.h**
- Library file: null

[Note]

This API must be called after the scene-auto module is initialized.

[Example]

scene_sample.c

HI_SCENE_Deinit

[Description]

Deinitializes the scene-auto module.

[Syntax]

```
HI_S32 HI_SCENE_Deinit(HI_VOID)
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success
Other value	Failure. For details, see section 5 " Error Code ."

[Requirement]

- Header file: **hi_scene.h**
- Library file: null



[Note]

This API must be called after the scene-auto module is initialized.

[Example]

scene_sample.c



4 Data Structure

The data structures are as follows:

- [HI_SCENE_PARAM_S](#): Defines the set of the scene-auto debugging parameters.
- [HI_SCENE_MODE_S](#): Defines the scene-auto media parameters.
- [HI_SCENE_PIPE_MODE_E](#): Defines the mode of the current media channel.
- [HI_SCENE_PIPE_ATTR_S](#): Defines the scene-auto media parameters for a pipe.
- [HI_SCENE_PIPE_TYPE_E](#): Defines the pipe type.

HI_SCENE_PARAM_S

[Description]

Defines the set of the scene-auto debugging parameters, including all the scene-auto debugging parameters.

[Syntax]

```
typedef struct hiSCENE_PARAM_S
{
    HI_SCENE_PIPE_PARAM_S astPipeParam[HI_SCENE_PIPE_TYPE_NUM];
} HI_SCENE_PARAM_S;
```



NOTE

The macros are defined as follows:

```
#define HI_SCENE_PIPE_TYPE_NUM (10)
```

[Member]

Member	Description
astPipeParam[HI_SCENE_PIPE_TYPE_NUM]	Array of the scene-auto debugging parameters. A group of debugging parameters correspond to one item in the array.

[Chip Difference]

None



[Note]

When the debugging parameters of different sensors are different, the internal members of `HI_SCENE_PIPE_PARAM_S` are also different.

[See Also]

[HI_SCENE_SetSceneMode](#)

HI_SCENE_MODE_S

[Description]

Defines the scene-auto media parameters. After an external media channel is set up, this parameter can inform the scene-auto module on which pipe the scene-auto function will take effect, so that the scene-auto debugging parameters can be configured for the selected pipe.

[Syntax]

```
typedef struct hiSCENE_MODE_S
{
    HI_SCENE_PIPE_MODE_E enPipeMode;
    HI_SCENE_PIPE_ATTR_S astPipeAttr[HI_SCENE_PIPE_MAX_NUM];
} HI_SCENE_MODE_S;
```

**NOTE**

The macros are defined as follows:

```
#define HI_SCENE_PIPE_MAX_NUM    (8)
```

[Member]

Member	Description
enPipeMode	Mode of the current media channel: linear, WDR, or HDR
astPipeAttr[HI_SCENE_PIPE_MAX_NUM]	Scene-auto media parameters for the pipes of the current media channel

[Chip Difference]

None

[Note]

- [HI_SCENE_PARAM_SHI_SCENE_SetSceneMode](#) For the media channel, when the linear mode is switched to the WDR mode or the resolution is changed, you need to reconfigure the scene-auto parameters. After configuring [HI_SCENE_PARAM_S](#), you can call [HI_SCENE_SetSceneMode](#) to complete the function.
- For details, see the sample.

[See Also]

- [HI_SCENE_SetSceneMode](#)
- [HI_SCENE_GetSceneMode](#)



- [HI_SCENE_PIPE_MODE_E](#)
- [HI_SCENE_PIPE_ATTR_S](#)
- HI_SCENE_PIPE_MAX_NUM

HI_SCENE_PIPE_MODE_E

[Description]

Defines the mode of the current media channel: linear, WDR, or HDR.

[Syntax]

```
typedef enum hiSCENE_PIPE_MODE_E
{
    HI_SCENE_PIPE_MODE_LINEAR = 0,
    HI_SCENE_PIPE_MODE_WDR,
    HI_SCENE_PIPE_MODE_HDR,
    HI_SCENE_PIPE_MODE_BUTT
} HI_SCENE_PIPE_MODE_E;
```

[Member]

Member	Description
HI_SCENE_PIPE_MODE_LINEAR	Linear mode
HI_SCENE_PIPE_MODE_WDR	WDR mode
HI_SCENE_PIPE_MODE_HDR	HDR mode
HI_SCENE_PIPE_MODE_BUTT	Invalid

[Chip Difference]

None

[Note]

- The exposure value varies according to the mode. The parameters and modules to be configured also vary according to the mode.
- For details, see the sample.

[See Also]

[HI_SCENE_MODE_S](#)

HI_SCENE_PIPE_ATTR_S

[Description]

Defines the scene-auto media parameters for a pipe.

[Syntax]

```
typedef struct hiSCENE_PIPE_ATTR_S
```



```
{  
    HI_BOOL bEnable;  
    HI_HANDLE MainPipeHdl;  
    HI_HANDLE VcapPipeHdl;  
    HI_HANDLE PipeChnHdl;  
    HI_HANDLE VencHdl;  
    HI_HANDLE VpssHdl;  
    HI_HANDLE VPortHdl;  
    HI_U8 u8PipeParamIndex;  
    HI_SCENE_PIPE_TYPE_E enPipeType;  
} HI_SCENE_PIPE_ATTR_S;
```

[Member]

Member	Description
bEnable	Whether the pipe is enabled The value true indicates that the pipe is enabled, and the value false indicates that the pipe is disabled.
MainPipeHdl	Pipe of the primary ISP, which is in the same group with the current pipe When MainPipeHdl is inconsistent with VcapPipeHdl , the ISP of MainPipeHdl controls the sensor. The sensor data is received by MainPipeHdl and VcapPipeHdl . However, the ISP of VcapPipeHdl performs image processing only and does not control the sensor.
VcapPipeHdl	Handle to the current pipe
PipeChnHdl	Pipe channel connected to the current pipe
VpssHdl	Handle to the video processing subsystem (VPSS) group connected to the current pipe
VPortHdl	Handle to the channel of the VPSS group connected to the current pipe
VencHdl	Handle to the video encoding (VENC) module connected to the current pipe
u8PipeParamIndex	Scene-auto parameters to be configured on the current pipe The value is the subscripts of the <code>astPipeParam[HI_SCENE_PIPE_TYPE_NUM]</code> array in <code>HI_SCENE_PARAM_S</code> .
enPipeType	Type of the current pipe

[Chip Difference]

None

[Note]



Currently, only one pipe channel, one VPSS, and one VENC module connected to the pipe can be dynamically adjusted, which are sufficient to cover all the service scenes. You may adjust the handle count as required.

[See Also]

[HI_SCENE_MODE_S](#)

HI_SCENE_PIPE_TYPE_E

[Description]

Defines the pipe type, indicating whether the pipe is used for photographing or video recording.

[Syntax]

```
typedef enum hiSCENE_PIPE_TYPE_E
{
    HI_SCENE_PIPE_TYPE_SNAP = 0,
    HI_SCENE_PIPE_TYPE_VIDEO,
    HI_SCENE_PIPE_TYPE_BUTT
} HI_SCENE_PIPE_TYPE_E;
```

[Member]

Member	Description
HI_SCENE_PIPE_TYPE_SNAP	The pipe is used for photographing.
HI_SCENE_PIPE_TYPE_VIDEO	The pipe is used for video recording.
HI_SCENE_PIPE_TYPE_BUTT	Invalid

[Chip Difference]

None

[Note]

None

[See Also]

[HI_SCENE_PIPE_ATTR_S](#)



5 Error Code

The scene-auto module provides the following error codes, as shown in [Table 5-1](#).

Table 5-1 Error codes provided by the scene-auto module

Error Code	Macro Definition	Description
0x80000001	HI_SCENE_EINVAL	The parameter is invalid.
0x80000002	HI_SCENE_ENOTINIT	The module is not initialized.
0x80000003	HI_SCENE_ENONPTR	The parameter contains a null pointer.
0x80000004	HI_SCENE_EOUTOFRANGE	The parameter value is beyond the range.
0x80000005	HI_SCENE_EINTER	An internal error occurs.
0x80000006	HI_SCENE_EINITIALIZED	The module is initialized.
-1	HI_FAILURE	The operation fails.



6 Parameter Description

6.1 ISP Parameters

The Hi3516C V500 ISP supports multiple pipes. Therefore, the ISP parameters are distinguished by setting the parameter **vi_pipe**. The ISP parameters are configured accordingly when **vi_pipe** is set to **0**, **1**, **2**, or **3**. Currently, the parameters related to the ISP are classified into static parameters and dynamic parameters. The static parameters do not require the association based on the ISO, exposure value, or exposure time ratio. However, the dynamic parameters require the association based on the ISO, exposure value, or exposure time ratio.

- In linear mode, the dynamic parameters require the association with the exposure value are as follows:
 - AE target value and AE offset
 - Gamma curve
 - LDCI enabling
 - Manual dehaze strength
 - Mesh shading strength

Dynamic parameters require the association based on the ISO:

- 3DNR parameters use manual interfaces. Therefore, interpolation is manually performed in the scene-auto module based on ISO association.
- The BayerNR random noise reservation strength table is manually interpolated based on the ISO within the scene-auto mode.

- In WDR mode, the dynamic parameters require the association with the exposure value are as follows:
 - AE target value and AE offset
 - Gamma curve
 - LDCI enabling
 - Manual dehaze strength
 - Mesh shading strength
 - DRC user-defined curve

Dynamic parameters require the association based on the ISO:

- Currently, 3DNR parameters use manual interfaces. Therefore, interpolation is manually performed in the scene-auto module based on ISO association.



- The BayerNR random noise reservation strength table is manually interpolated based on the ISO within the scene-auto mode.
- For details about DRC parameters such as **LocalMixingBrightMax** and **LocalMixingBrightMin**, see [Table 6-13](#).

6.1.1 AE Parameters

The AE module provides static parameters and dynamic parameters. The static parameters are used to configure the AE tolerance value, AE statistics weight table, AE extended allocation route table, AE convergence speed, and maximum system gain. The dynamic parameters are used to configure the AE target value and AE offset, which are associated with the exposure value.

[Table 6-1](#) describes the static parameters of the AE module.

Table 6-1 Static parameters of the AE module

Parameter	Description	Value Range
AERouteExValid	Enabling of the extended AE allocation route. If this parameter is set to HI_TRUE , the extended allocation route is enabled. Otherwise, the common AE allocation route is used.	[0, 1]
AERunInterval	Running interval of the AE algorithm. The value 1 indicates that the AE algorithm runs every frame, the value 2 indicates that the AE algorithm runs once every two frames. The same rule applies. It is recommended that the configured value be less than or equal to 2 . Otherwise, the AE auto speed is affected. In WDR mode, the recommended value is 1 for smoother AE convergence.	[1, 255]
AutoSpeed	Default initial AE auto speed. The HiSilicon AE algorithm uses the value of this parameter as the auto speed of the initial 100 frames. This parameter can be used to accelerate the startup for motion DV. You are advised to set this member to 128 for the product form that concerns fast boot. The default value 0 is used if this parameter is left unconfigured. The AE algorithm limits the value range of this parameter to [64, 255] internally.	[0, 255]
AutoTolerance	AE auto tolerance value	[0, 255]
AutoBlackDelayFrame	AE adjustment starts when the image luminance is less than the target luminance for u16BlackDelayFrame frames.	[0, 65535]
AutoWhiteDelayFrame	AE adjustment starts when the image luminance is greater than the target luminance for u16WhiteDelayFrame frames.	[0, 65535]
AutoSysGainMax	System gain range (10-bit decimal precision)	[0x400, 0xFFFFFFFF]



Parameter	Description	Value Range
		The specific range is sensor-related.
AutoExpTimeMax	Maximum exposure time	(0x0, 0xFFFFFFFF) The specific range is sensor-related.

Table 6-2 describes the dynamic parameters of the AE module.

Table 6-2 Dynamic parameters of the AE module

Parameter	Description	Value Range
aeExpCount	Luminance adjustment based on the illuminance. The value is the group count.	[0, 8]
aeExpLtoHThresh	Threshold of the exposure level from bright to dark when the illuminance changes from bright to dark	[Minimum exposure time (μ s) x Multiple of the minimum system gain, Maximum exposure time (μ s) x Multiple of the maximum system gain]
aeExpHtoLThresh	Threshold of the exposure level from dark to bright when the illuminance changes from dark to bright	[Minimum exposure time (μ s) x Multiple of the minimum system gain, Maximum exposure time (μ s) x Multiple of the maximum system gain]
AutoCompensation	Target AE luminance value. The recommended value range is [0x38, 0x40].	[0, 255]
AutoHistOffset	Maximum influence on the average statistics exerted by ROIs Default value: 0x10	[0, 255]

6.1.2 Global CAC Parameters

The global chromatic aberration correction (CAC) module provides only the static parameters.

Table 6-3 describes the static parameters of the global CAC module.

Table 6-3 Static parameters of the global CAC module

Parameter	Description	Value Range
GlobalCacEnable	Lateral CAC enable	[0, 1]
VerCoordinate	Vertical coordinate of the optical center	[0, Height] Height is the height of the image processed by the ISP.



Parameter	Description	Value Range
HorCoordinate	Horizontal coordinate of the optical center.	[0, Width] Width is the width of the image processed by the ISP.
ParamRedA	Coefficient a of the radius polynomial corresponding to the R channel	[−256, +255]
ParamRedB	Coefficient b of the radius polynomial corresponding to the R channel	[−256, +255]
ParamRedC	Coefficient c of the radius polynomial corresponding to the R channel	[−256, +255]
ParamBlueA	Coefficient a of the radius polynomial corresponding to the B channel	[−256, +255]
ParamBlueB	Coefficient b of the radius polynomial corresponding to the B channel	[−256, +255]
ParamBlueC	Coefficient c of the radius polynomial corresponding to the B channel	[−256, +255]
VerNormShift	Normalization shift in the vertical direction	[0, 7]
VerNormFactor	Normalization factor in the vertical direction	[0, 31]
HorNormShift	Normalization shift in the horizontal direction	[0, 7]
HorNormFactor	Normalization factor in the horizontal direction	[0, 31]
CorVarThr	Variance threshold of lateral CAC	[0, 4095]

6.1.3 Saturation Parameters

Currently, saturation parameters are all static parameters.

Table 6-4 describes the static parameters of the saturation module.

Table 6-4 Static parameters of the saturation module

Parameter	Description	Value Range
AutoSat	Saturation LUT, associated with the ISO	[0, 255]

6.1.4 LDCI Parameters

The CMOS driver of the sensor configures the parameters provided by the LDCI module. Because of the scene-auto function, the LDCI parameters are enabled or disabled based on the exposure value.

Table 6-5 describes the dynamic parameters of the LDCI module.

**Table 6-5** Static parameters of the LDCI module

Parameter	Description	Value Range
EnableCount	LDCI enable based on different exposure values. The value is the group count.	[0, 6]
EnableExpThreshLtoH	Threshold of the exposure level from bright to dark when the illuminance changes from bright to dark	[Minimum exposure time (μs) x Multiple of the minimum system gain, Maximum exposure time (μs) x Multiple of the maximum system gain]
Enable	LDCI module enable	[0, 1]

6.1.5 Dehaze Parameters

The Dehaze module provides static parameters and dynamic parameters. The static parameters are mainly used to configure the basic dehaze attributes, such as configuration enabling, customized curve enabling, manual or auto dehaze selection, and curve table customization. For the dynamic parameters, the dehaze strength is associated with the exposure value.

[Table 6-6](#) describes the static parameters of the Dehaze module.

Table 6-6 Static parameters of the Dehaze module

Parameter	Description	Value Range
bEnable	Dehaze enable	[0, 1]
bDehazeUserLutEnable	Enabling of the user-defined dehaze curve	[0, 1]
DehazeOpType	Operating mode	[0, 1]
DehazeLut[256]	Enabling of the user-defined dehaze curve lookup table (LUT)	[0, 255]

[Table 6-7](#) describes the dynamic parameters of the Dehaze module.

Table 6-7 Dynamic parameters of the Dehaze module

Parameter	Description	Value Range
ExpThreshCnt	Manual dehaze strength based on the illuminance. The value is the group count.	[0, 5]
ExpThreshLtoH	Threshold of the exposure level from bright to dark when the illuminance changes from bright to dark	[Minimum exposure time (μs) x Multiple of the minimum system gain, Maximum exposure time (μs) x Multiple of the maximum system gain]



Parameter	Description	Value Range
ManualDehazeStr	Manual dehaze strength	[0, 255]

6.1.6 Mesh Shading Parameters

The Mesh Shading module provides static parameters and dynamic parameters. The static parameters are mainly used for configuration enabling. For the dynamic parameters, the shading correction strength is associated with the exposure value.

Table 6-8 describes the static parameters of the Mesh Shading module.

Table 6-8 Static parameters of the Mesh Shading module

Parameter	Description	Value Range
bEnable	Mesh shading correction enable	[0, 1]

Table 6-9 describes the dynamic parameters of the Mesh Shading module.

Table 6-9 Dynamic parameters of the Mesh Shading module

Parameter	Description	Value Range
ExpThreshCnt	Manual dehaze strength based on the illuminance. The value is the group count.	[0, 5]
ExpThreshLtoH	Threshold of the exposure level from bright to dark when the illuminance changes from bright to dark	[Minimum exposure time (μs) x Multiple of the minimum system gain, Maximum exposure time (μs) x Multiple of the maximum system gain]
ManualStrength	Global mesh shading correction strength after calibration. When the lens shading is serious, the compensation gain of the four corners of the image is large. As a result, the noise is obvious and grids may appear. If the value of u16MeshStrength is less than 4096 , the compensation value of mesh shading correction can be reduced. In this way, the compensation gain of the four corners is reduced to mitigate the noise and grid effects.	[0, 65535]

6.1.7 WDRExposureAttr Parameters

The WDRExposureAttr module provides static parameters and dynamic parameters. The static parameters can be used to set the exposure ratio type and manual exposure ratio. The dynamic parameters can be used to set the upper limit and lower limit of the exposure ratio, which need to be associated with the exposure value.



Table 6-10 describes the static parameters of the WDRExposureAttr module.

Table 6-10 Static parameters of the WDRExposureAttr module

Parameter	Description	Value Range
ExpRatioType	This parameter is valid only in multi-frame combination WDR mode. OP_TYPE_AUTO : Automatically calculates the exposure time ratio of the long frame to the short frame based on scenarios. OP_TYPE_MANUAL : Manually configures the exposure time ratio of the long frame to the short frame.	[0, 1]
ExpRatioMax	This parameter is valid only in multi-frame combination WDR mode. When enExpRatioType is OP_TYPE_AUTO , u32ExpRatioMin indicates the maximum exposure time ratio of the longest frame to the shortest frame. That is, the maximum S/VS exposure time ratio in 2-frame combination mode, the maximum S/VS exposure time ratio in 3-frame combination mode, or the maximum L/VS exposure time ratio in 4-frame combination mode. When enExpRatioType is OP_TYPE_MANUAL , u32ExpRatioMax is invalid. The precision of u32ExpRatioMax is 6-bit decimal precision. The value 0x40 indicates 1x exposure time ratio.	[0x40, 0x4000]
ExpRatioMin	This parameter is valid only in multi-frame combination WDR mode. When enExpRatioType is OP_TYPE_AUTO , u32ExpRatioMin indicates the minimum exposure time ratio of the longest frame to the shortest frame. When enExpRatioType is OP_TYPE_MANUAL , u32ExpRatioMin is invalid. The format is unsigned 6.6-bit fixed point. The value 0x40 indicates 1x exposure time ratio. The default value is 0x40 .	[0x40, u32ExpRatioMax]
ExpRatio	This parameter is valid only in multi-frame combination WDR mode. When enExpRatioType is OP_TYPE_AUTO , au32ExpRatio is invalid. When enExpRatioType is OP_TYPE_MANUAL , au32ExpRatio indicates the expected exposure time ratio of two adjacent frames in multi-frame combination WDR mode and can be read and written. au32ExpRatio[0] indicates the S/VS exposure time ratio, au32ExpRatio[1] indicates the M/S exposure time ratio, and au32ExpRatio[2] indicates the L/M exposure time ratio.	[0x40, 0xFFFF]

**NOTE**

VS, S, M, and L represent the exposure time of the extremely short frame, the short frame, the middle frame, and the long frame, respectively. Only VS and S are valid in 2-frame combination mode. Only VS, S, and M are valid in 3-frame combination mode. VS, S, M, and L are all valid in 4-frame combination mode. The precision of **u32ExpRatioMax** is 6-bit decimal precision. The value **0x40** indicates 1x exposure time ratio.

6.1.8 FSWDR Parameters

The FSWDR module provides only dynamic parameters. The dynamic parameters are used to configure the motion compensation enabling, NR mode of the combination module, and NR strength of the combination module.

6.1.9 Gamma Parameters

In different scenarios, the contrast values are different, and the requirements are different. Therefore, different gamma values need to be set according to different exposure values.

Table 6-11 describes the Gamma parameters.

Table 6-11 Gamma parameters

Parameter	Description	Value Range
gammaInterval	Gamma switchover rate	[1, 4294967295] Indicates the number of times that the previous gamma curve is converted to the next gamma curve.
gammaExpCount	Exposure level count	[1, 10]
gammaExpThreshLtoH	Threshold of the exposure level from bright to dark when the illuminance changes from bright to dark	[Minimum exposure time (μs) x Multiple of the minimum system gain, Maximum exposure time (μs) x Multiple of the maximum system gain]
gammaExpThreshHtoL	Threshold of the exposure level from dark to bright when the illuminance changes from dark to bright	[Minimum exposure time (μs) x Multiple of the minimum system gain, Maximum exposure time (μs) x Multiple of the maximum system gain]
Table_K	Gamma value of group K	[1, 4095]

6.1.10 DRC Parameters

The DRC module provides static parameters and dynamic parameters. The static parameters are mainly used to configure the DRC curve type and DRC strength. The dynamic parameters in linear mode and WDR mode are provided separately. The dynamic parameters in linear mode mainly include the manual strength and curve parameters. The dynamic parameters in WDR mode include the detail enhancement (DE) parameters, such as

LocalMixingBrightMax, **LocalMixingBrightMin**, **u8LocalMixingDarkMax**, and **u8LocalMixingDarkMin**, and the DRC user-defined curve. For Hi3516C V500, two groups of user-defined DRC curves **DRCMappingValue1** and **DRCMappingValue2** are



provided. Interpolation is performed for the combination of the two groups of curves based on the exposure ratio. Different user-defined DRC curves are used for strong WDR scenarios and weak WDR scenarios.

Table 6-12 and Table 6-13 describe the DRC parameters.

Table 6-12 DRC parameters in linear mode

Parameter	Description	Value Range
Enable	DRC enable in linear mode	[0, 1]
IsoCnt	ISO count	[0, 10]
IsoLevel	Threshold for grading ISO values	[Minimum system ISO, Maximum system ISO] The specific range is sensor-related.
LocalMixingBrightMax	Maximum gain of the positive detail	[0x0, 0x80]
LocalMixingBrightMin	Minimum gain of the positive detail	[0x0, 0x40]
u8LocalMixingDarkMax	Maximum gain of the negative detail	[0x0, 0x80]
u8LocalMixingDarkMin	Minimum gain of the negative detail	[0x0, 0x40]
u8BrightGainLmt	Target value of the luminance gain limit of the bright region. A larger value indicates a greater restriction.	[0x0, 0xF] For details, see the <i>HiISP Development Reference</i> .
u8BrightGainLmtStep	Adaptive step of the luminance gain limit of the bright region. A smaller value indicates a greater restriction.	[0x0, 0xF] For details, see the <i>HiISP Development Reference</i> .
u8DarkGainLmtY	Luminance gain limit of the dark region. A larger value indicates a greater restriction.	[0x0, 0x85] For details, see the <i>HiISP Development Reference</i> .
u8DarkGainLmtC	Chrominance gain limit of the dark region. A larger value indicates a greater restriction.	[0x0, 0x85] For details, see the <i>HiISP Development Reference</i> .
u8ContrastControl	Global contrast control The parameter effect is associated with the luminance distribution of the image. The recommended value range is [6, 10].	[0x0, 0xF] For details, see the <i>HiISP Development Reference</i> .
DetailAdjustFactor	Coefficient of detail fine tuning. A larger value indicates stronger overall details.	[-0xF, 0xF] For details, see the <i>HiISP Development Reference</i> .
Asymmetry	Asymmetry curve generation factor. A smaller value indicates greater stretching in dark regions.	[0x1, 0x1E] For details, see the <i>HiISP Development Reference</i> .
SecondPole	Asymmetry curve generation factor. A larger value indicates greater stretching for the overall	[0x96, 0xD2] For details, see the <i>HiISP Development Reference</i> .



Parameter	Description	Value Range
	luminance.	<i>Development Reference.</i>
Compress	Asymmetry curve generation factor. The larger the value, the more the curve is compressed in the vertical direction and the lower the overall luminance.	[0x64, 0xC8] For details, see the <i>HiISP Development Reference.</i>
Stretch	Asymmetry curve generation factor. The smaller the value, the higher the overall luminance.	[0x1E, 0x3C] For details, see the <i>HiISP Development Reference.</i>
Strength	DRC strength in manual mode. The larger the value, the higher the overall luminance.	[0x0, 0x3FF] For details, see the <i>HiISP Development Reference.</i>

Table 6-13 DRC parameters in WDR mode

Parameter	Description	Value Range
RationCnt	Exposure ratio count	[0, 6]
RationLevel	Threshold for grading exposure ratios	This parameter is used in WDR mode. [1, 256]
ExpCnt	Number of different exposure durations	[0, 7]
ExpThreshLtoH	Threshold for grading the exposure durations in descending order.	[Minimum exposure time (μs) x Multiple of the minimum system gain, Maximum exposure time (μs) x Multiple of the maximum system gain]
Alpha	Interpolation ratio of two DRC curves	[0, 255]
IsoCnt	ISO value count	[0, 7]
IsoLevel	Threshold for grading ISO values	[Minimum system ISO, Maximum system ISO] The specific range is sensor-related.
Interval	DRC parameter conversion speed	[0, 4294967295]
LocalMixingBrightMax	Maximum gain of the positive detail	[0x0, 0x80]
LocalMixingBrightMin	Minimum gain of the positive detail	[0x0, 0x40]
u8LocalMixingDarkMax	Maximum gain of the negative detail	[0x0, 0x80]
u8LocalMixingDarkMin	Minimum gain of the negative detail	[0x0, 0x40]
u8DarkGainLmtY	Luminance gain limit of the dark region. A larger value indicates a greater restriction.	[0x0, 0x85] For details, see the <i>HiISP Development Reference.</i>



Parameter	Description	Value Range
u8DarkGainLmtC	Chrominance gain limit of the dark region. A larger value indicates a greater restriction.	[0x0, 0x85] For details, see the <i>HiISP Development Reference</i> .
DetailAdjustFactor	Factor of detail fine-tuning. A larger value indicates weaker overall detail. A smaller value indicates stronger overall detail.	[-0xF, +0xF] For details, see the <i>HiISP Development Reference</i> .
SpatialFltCoef	Spatial filtering coefficient. A larger value indicates less obvious motion halo and a smaller amount of detail. A smaller value indicates better detail and more obvious motion halo.	[0x0, 0xA] For details, see the <i>HiISP Development Reference</i> .
u8RangeFltCoef	Range filtering coefficient. A smaller value indicates less obvious halo but more possible borders at the strong edges.	[0x0, 0xA] For details, see the <i>HiISP Development Reference</i> .
GradRevMax	Border correction strength. A larger value indicates that the border is weakened more obviously, but may cause detail loss.	[0, 0xFF] For details, see the <i>HiISP Development Reference</i> .
GradRevThr	Border detection threshold. A larger value indicates that the border is weakened more obviously, but may cause detail loss.	[0, 0xFF] For details, see the <i>HiISP Development Reference</i> .
Compress	It is used to generate the Asymmetry curve. The larger the value, the more the curve is compressed in the vertical direction and the lower the overall luminance.	[0x64, 0xC8] For details, see the <i>HiISP Development Reference</i> .
Stretch	It is used to generate the Asymmetry curve. The smaller the value, the higher the overall luminance.	[0x1E, 0x3C] For details, see the <i>HiISP Development Reference</i> .

6.1.11 3DNR Parameters

Currently, the 3DNR interface provides static parameters and uses the 3DNR manual interface. The 3DNR parameters of each ISO must be configured. Interpolation is performed in the scene-auto module based on the ISO value.

For details about the 3DNR parameters, see the *Description of 3DNR Parameter Configurations*.



6.1.12 Sharpen Parameters

Currently, the sharpen parameters are all static parameters, which are automatically obtained based on the ISO interpolation. The texture sharpening strength table and edge sharpening strength table need to be interpolated based on the ISO value within the scene-auto mode.

Table 6-14 describes the static parameters of the sharpen module.

Table 6-14 Parameters of the sharpen module

Parameter	Description	Value Range
u8OverShoot	Overshoot strength. Overshoot indicates the white points and white borders after sharpening.	[0,127] For details, see the <i>HiISP Development Reference</i> .
u8UnderShoot	Undershoot strength. Undershoot indicates the black points and black borders after sharpening.	[0, 127] For details, see the <i>HiISP Development Reference</i> .
TextureStrTable	Sharpen strength for directionless details and textures	[0, 4095] For details, see the <i>HiISP Development Reference</i> .
EdgeStrTable	Sharpening strength for directional edges	[0, 4095] For details, see the <i>HiISP Development Reference</i> .

6.1.13 Demosaic Parameters

Currently, the demosaic parameters are all static parameters, which are automatically obtained based on the ISO interpolation.

Table 6-15 describes the static parameters of the demosaic module.

Table 6-15 Parameters of the demosaic module

Parameter	Description	Value Range
NonDirMFDetailEhcStr	Non-directional IF texture enhancement strength. A larger value indicates stronger enhancement of non-directional IF texture detail and noise.	[0,127] For details, see the <i>HiISP Development Reference</i> .

6.1.14 BayerNR Parameters

Currently, the demosaic parameters are all dynamic parameters, which are associated with the ISO. The random noise reservation strength table needs to be interpolated based on the ISO value within the scene-auto mode.

Table 6-16 describes the static parameters of the BayerNR module.

**Table 6-16** BayerNR Parameters

Parameter	Description	Value Range
IsoCnt	ISO count	[0, 16]
IsoLevel	Threshold for grading ISO values	[Minimum system ISO, Maximum system ISO] The specific range is sensor-related.
CoringWgt	Reservation strength of the random noise. A larger value indicates more random noises are reserved.	[0, 3200] For details, see the <i>HiISP Development Reference</i> .
FineStr	Luminance noise reduction strength. A larger value indicates greater NR strength.	[0, 128] For details, see the <i>HiISP Development Reference</i> .
CoringRatioTable	Noise reservation strength based on the luminance. A larger value indicates more luminance noise is reserved.	[0,1023] For details, see the <i>HiISP Development Reference</i> .

6.1.15 Detail Parameters

Currently, the detail parameters are all static parameters, which are associated with the ISO.

[Table 6-17](#) describes the static parameters of the detail module.

Table 6-17 Parameters of the detail module

Parameter	Description	Value Range
GlobalGain	Gain of the global overlay strength	[0, 256] For details, see the <i>HiISP Development Reference</i> .