Developer's Guide

AutoSDK

Version 2.5.x

Seenaptec

www.seenaptec.com

CONTENTS

1 definitions 8
2 autosdk overview 9
2.1 General information . . . 9
2.2 System requirements . . . 9
2.3 Product operation modes . . . 10
2.3.1 AutoSDK Lite (Parking) . . . 10
2.3.2 AutoSDK (Freeflow) . . . 10
2.4 Product licensing . . . 10
2.5 Product structure . . . 11
2.6 Installation (on Windows OS) . . . 13
3 first application 14
3.1 Loading the modules . . . 14
3.2 Creating the principal . . . 14
3.3 Configuring the principal . . . 15
3.4 Recognizing the license plates . . . 16
3.5 Processing the recognition results . . . 17
4 description of additional cli utilities 20
4.1 Description of the vpwfetch utility . . . 20
4.1.1 Brief overview . . . 20
4.1.2 Expression semantics . . . 20
4.1.3 Configuration . . . 24
4.1.4 Examples of commands . . . 27
4.2 Description of the lsvpwi utility . . . 27
4.2.1 Overview . . . 27
4.2.2 Options and examples . . . 27
4.3 Description of the lsvpwc utility . . . 30
4.3.1 Overview . . . 30
4.3.2 Options . . . 30
5 description of operations with aorp objects 32
5.1 VpwprincOpen . . . 32
5.1.1 Name . . . 32
5.1.2 Syntax . . . 32
5.1.3 Description . . . 32
5.1.4 Return values . . . 33
5.2 AorpRetain . . . 34
5.2.1 Name . . . 34
5.2.2 Syntax . . . 34
5.2.3 Description . . . 34
5.2.4 Return values . . . 34
5.3 AorpRelease . . . 35

5.3.1 Name . . . 35
5.3.2 Syntax . . . 35
5.3.3 Description . . . 35
5.3.4 Return values . . . 35
5.4 VodiprincGetparam . . . 36
5.4.1 Name . . . 36
5.4.2 Syntax . . . 36
5.4.3 Description . . . 36
5.4.4 Return values . . . 36
5.5 VodiprincSetparam . . . 37
5.5.1 Name . . . 37
5.5.2 Syntax . . . 37
5.5.3 Description . . . 37
5.5.4 Return values . . . 41
5.6 VodiprincApplyparam . . . 42
5.6.1 Name . . . 42
5.6.2 Syntax . . . 42
5.6.3 Description . . . 42
5.6.4 Return values . . . 42
5.7 VodiprincControl . . . 43
5.7.1 Name . . . 43
5.7.2 Syntax . . . 43
5.7.3 Description . . . 43
5.7.4 Return values . . . 43
5.8 VodiprincProcess . . . 45
5.8.1 Name . . . 45
5.8.2 Syntax . . . 45
5.8.3 Description . . . 45
5.8.4 Return values . . . 46
5.9 VodiprincFlush . . . 47
5.9.1 Name . . . 47
5.9.2 Syntax . . . 47
5.9.3 Description . . . 47
5.9.4 Return values . . . 47
5.10 VodiensCount . . . 48
5.10.1 Name . . . 48
5.10.2 Syntax . . . 48
5.10.3 Return values . . . 48
5.11 VodiensClear . . . 49
5.11.1 Name . . . 49
5.11.2 Syntax . . . 49
5.11.3 Description . . . 49
5.11.4 Return values . . . 49
5.12 VodiensAdd . . . 50
5.12.1 Name . . . 50
5.12.2 Syntax . . . 50

5.12.3 Description . . . 50
5.12.4 Return values . . . 50
5.13 VodiensRemove . . . 51
5.13.1 Name . . . 51
5.13.2 Syntax . . . 51
5.13.3 Description . . . 51
5.13.4 Return values . . . 51
5.14 VodiensDelete . . . 52
5.14.1 Name . . . 52
5.14.2 Syntax . . . 52
5.14.3 Description . . . 52
5.14.4 Return values . . . 52
5.15 VodiensGet . . . 53
5.15.1 Name . . . 53
5.15.2 Syntax . . . 53
5.15.3 Description . . . 53
5.15.4 Return values . . . 53
5.16 VodiensUnique . . . 54
5.16.1 Name . . . 54
5.16.2 Syntax . . . 54
5.16.3 Description . . . 54
5.16.4 Return values . . . 54
5.17 VodiensCombine . . . 55
5.17.1 Name . . . 55
5.17.2 Syntax . . . 55
5.17.3 Description . . . 55
5.17.4 Return values . . . 55
5.18 VodiensCombine_v2 . . . 56
5.18.1 Name . . . 56
5.18.2 Syntax . . . 56
5.18.3 Description . . . 56
5.18.4 Return values . . . 56
5.19 VodiresGetuserdata . . . 57
5.19.1 Name . . . 57
5.19.2 Syntax . . . 57
5.19.3 Description . . . 57
5.19.4 Return values . . . 57
5.20 VodiresSetuserdata . . . 58
5.20.1 Name . . . 58
5.20.2 Syntax . . . 58
5.20.3 Description . . . 58
5.20.4 Return values . . . 58
5.21 VodiresFetchinfo . . . 59
5.21.1 Name . . . 59
5.21.2 Syntax . . . 59
5.21.3 Description . . . 59

5.21.4 Return values . . . 59
5.22 VodiResultInfoDestroy . . . 60
5.22.1 Name . . . 60
5.22.2 Syntax . . . 60
5.22.3 Description . . . 60
6 description of structures 61
6.1 vodi_vpw_country_id . . . 61
6.1.1 Name . . . 61
6.1.2 Syntax . . . 61
6.1.3 Description . . . 61
6.2 vodi_image . . . 62
6.2.1 Name . . . 62
6.2.2 Syntax . . . 62
6.2.3 Description . . . 62
6.3 vodi_vpw_options . . . 63
6.3.1 Name . . . 63
6.3.2 Syntax . . . 63
6.3.3 Description . . . 63
6.4 vodi_ucontext . . . 64
6.4.1 Name . . . 64
6.4.2 Syntax . . . 64
6.4.3 Description . . . 64
6.5 vodi_plate_info_spec . . . 65
6.5.1 Name . . . 65
6.5.2 Syntax . . . 65
6.5.3 Description . . . 66
6.6 vodi_plate . . . 68
6.6.1 Name . . . 68
6.6.2 Syntax . . . 68
6.6.3 Description . . . 68
6.7 aorp_error . . . 70
6.7.1 Name . . . 70
6.7.2 Syntax . . . 70
6.7.3 Description . . . 70
7 description of operations with templates 71
7.1 VodiISO3166children . . . 71
7.1.1 Name . . . 71
7.1.2 Syntax . . . 71
7.1.3 Description . . . 71
7.1.4 Return values . . . 72
7.2 LpvlibShowPlateid . . . 73
7.2.1 Name . . . 73
7.2.2 Syntax . . . 73
7.2.3 Description . . . 73
7.2.4 Return values . . . 73
7.3 LpvlibReadPlateid . . . 74

7.3.1 Name . . . 74
7.3.2 Syntax . . . 74
7.3.3 Description . . . 74
7.3.4 Return values . . . 74
7.4 LpvlibModulesOf . . . 75
7.4.1 Name . . . 75
7.4.2 Syntax . . . 75
7.4.3 Description . . . 75
7.4.4 Return values . . . 76
8 description of grammar for specifying the templates sets 77

1

DEFINITIONS

- LP — License Plate
- LPR — License Plate Recognition

- **Principal** (**Vpwprinc**) is an object for which the LPR operations (**VodiprincProcess**) are defined. One instance of principal is equivalent to one instance of recognition channel. Analysis of one videostream can be devided into several recognition threads in order to balance the CPU load while high-definition video processing.
- **Result** (**Vpwres**) is an object which represents a list of recognized LP variants (depending on its similarity to specific LP template).
- **Ensemble** (**Vpwens**) is an object which represents a list of LPR results.

  > **Vpwprinc** (principal), **Vpwens** (ensemble) and **Vpwres** (result) belong to a group of so-called AORP-objects.

- **AORP object** is a representative of many AORP objects. **Vpwprinc** (principal), **Vpwens** (ensemble) and **Vpwres** (result) objects belong to this category. Each AORP object has its own set of specific operations.
- **Shared library** is a library for general use, also known as dynamic library.
- **Archive library** is a library also known as static.
- **Stub library** is a library used on Windows platform for linking with shared libraries.

2

AUTOSDK OVERVIEW

2.1 general information

AutoSDK is a software development kit which contains the C/C++ runtime libraries and header files. The SDK is used to complement systems with functionality of license plate recognition. Below are the examples of such systems:

- traffic surveillance systems;
- parking management systems;
- perimeter protection & access control systems;
- electronic toll collection systems;
- intelligent transportation systems;
- safe city solutions;
- embedded devices, speed radars, cameras;
- weighing systems.

The SDK is to be used in application development and provides the following functionality:

- recognition of both single- and double-line license plates (LPs);
- real-time video stream analysis: frame processing time is within 10-100 ms depending on resolution;
- recognition of up to 20 plates in the same frame (which allows for multi-lane surveillance);
- maximum speed for the license plates to be recognized is 250 km/h (under proper LPR camera settings);
- determining the direction of vehicle movement;
- recognition fine tuning: LP templates selection, creation of multiple recognition zones (within the frame), control over parameters of resulting data;
- selecting the best recognition result for each plate and filtering out the duplicates.

2.2 system requirements

- **Supported OS:** Unix family (e.g. Mac OS X, Linux, FreeBSD) and Microsoft Windows family (e.g. 2000, 2003, XP, Vista, Seven, 8/8.1, 10).

- **CPU architectures:** x86 (x86-32, x86-64), ARMv7, ARMv8 AArch32.
- **CPU model:** selected individually (in accordance with number of recognition threads to be used, video framerate and resolution). When adding one or more recognition threads, CPU load increases linearly.
- **RAM:** 2Gb and more. Amount of memory used by one recognition thread depends on video framerate and resolution and is approximately equal to 200 Mb.
- **Programming languages:** C, C++.
- **Minimal height of license plate symbols in the frame (that allows for them to be recognized):**
  - 14-20 px for cameras with no hardware compression (analog, machine vision cameras);
  - 20-30 px for cameras with hardware compression (IP-cameras).
- **Optimal recognition camera angles:**
  - vertical — 18-20 (maximum — 30) degrees;
  - horizontal — 5-10 (maximum — 20) degrees.
- **Angle of a license plate on the image:** 5-10 degrees.
- **Input data:** uncompressed video stream (RAW, 8 bpp, grayscale).

2.3 product operation modes

2.3.1 AutoSDK Lite (Parking)

- Processes up to 6 fps.
- Required speed of vehicles in surveillance zone: up to 20 km/h

2.3.2 AutoSDK (Freeflow)

- Processes up to 25 fps.
- Required speed of vehicles in surveillance zone: up to 220 km/h

2.4 product licensing

Dynamic libraries of **AutoSDK** are protected and can be used only when appropriate license keys are installed locally[1] (Table 1).

[1] licensing protection of **AutoSDK** is carried out with the Sentinel HASP solution by Gemalto

Table 1: Features of lisence keys based on their types

Key type	Description
HASP HL Key	Provided as a hardware token (USB-key). This key is physically installed onto a server with the protected software to be used. The key may be transferred without losing the license.
HASP SL Key	Provided as a service. It is attached to hardware of the computer. If protected products are transferred to another server, a new license should be purchased.

The following features of **AutoSDK** depend on the type of purchased license:

- maximum number of active recognition threads;
- license plate templates (grouped by issuer countries) available during the recognition;
- maximum video processing speed (6 fps or 25 fps, see Product operation modes).

> To view information on **AutoSDK** licenses available locally on your computer, use the lsvpwc utility (see Description of the lsvpwc utility).

2.5 product structure

AutoSDK distribution package consists of file package and license key, which enables the operation of protected files. The package has the following structure:

- **_doc**:
  - AutoSDK_v2.5_RU.pdf, AutoSDK_v2.5_EN.pdf — developer's guides (in Russian and English);
  - AutoSDK_2.5.x_change_log_RU.pdf, AutoSDK_2.5.x_change_log_EN.pdf — **AutoSDK** change logs (in Russian and English);
  - **templates** — images of supported license plate templates.
- **_examples** — **AutoSDK** usage examples.
- **_redist** — redistributable software[2].
- **_opt**:
  - **include** — header files:
    - Vodi — **AutoSDK** header files;
    - Bo — header files and support libraries (e.g. for code portability);
    - ImageMagick — libraries for bitmap images processing.
  - **bin** [**Unix**] — utilities, usage examples;
  - **bin** [**Windows**] — utilities, dynamic libraries, e.g.:

[2] e.g. Sentinel LDK Run-time Environment, Microsoft Redistributable Package

    * vpwfetch — CLI utility for recognizing license numbers (see Description of the vpwfetch utility);
    * lsvpwi — CLI utility to print information on the currently available license plate templates (see Description of the lsvpwi utility);
    * lsvpwc — CLI utility to print information on available license keys (see Description of the lsvpwc utility);
    * Bo.dll — support library;
    * Bo_g.dll — debug version;
    * Vodi.dll — main library;
    * Vodi_g.dll — debug version.
  - **lib** [**Unix**] — object files, dynamic and archive libraries:
    * vpwfetch — CLI utility for recognizing license numbers (see Description of the vpwfetch utility);
    * lsvpwi — CLI utility to print information on the currently available license plate templates (see Description of the lsvpwi utility);
    * lsvpwc — CLI utility to print information on available license keys (see Description of the lsvpwc utility);
    * [PFX]Bo[SFX] — support library;
    * [PFX]Bo_g[SFX] — debug version;
    * [PFX]Vodi[SFX] — main library;
    * [PFX]Vodi_g[SFX] — debug version.
  - **lib** [**Windows**] — object files, static and stub libraries, e.g.:
    * Bo.lib — stub library for Bo.dll;
    * Bo_g.lib — debug version;
    * Vodi.lib — stub library for Vodi.dll;
    * Vodi_g.lib — debug version.
  - **libexec** — libraries with implicit linking.

As a part of a dynamic library name (for Unix), there may be:

- [PFX] — system-dependent prefix (for example, lib);
- [SFX] — system-dependent suffix containing the file version and its extension (for example, .so.2.5.0; 2.5.0.dylib, -2_5_0.dll).

The folders in **_doc/templates** directory contain images of LPs of various types. The LPs of each set were issued in the country stated in folder name. The LP images names are equivalent to order numbers of corresponding LP templates within the country module (**vpwi-[country-ID]**[a]).

[a] for example, the "vpwi-co" module, which is used for recognition of Colombian LPs

The **_doc/templates/templates_map.txt**[a] file describes the conformance between the former LP template identifiers used by **AutoSDK** and the new ones[b]. As a useful alternative able to provide more detailed information on each of the templates, use the lsvpwc CLI utility (see Description of the lsvpwc utility).

[a] available since 2.5.8 version
[b] available since 2.5.1 version

2.6 installation (on windows os)

To install **AutoSDK** properly, it is recommended to adhere to the following instruction:

> Let's assume INSTALLDIR is a product installation path.

1. Run the **setup.exe** file which is located in the root folder of **AutoSDK** file package. Step through the installation program. As a result, **AutoSDK** files will be relocated to INSTALLDIR.
2. Add path to INSTALLDIR\opt\include as a header search path for C/C++ preprocessor.
3. Add path to INSTALLDIR\opt\lib as a library search path for C/C++ linker.

   > For example, when using Microsoft Visual Studio, the paths are to be specified in Additional Include Directories section.

4. Assign path to the INSTALLDIR\opt\bin to PATH environment variable.
5. Assign path to the INSTALLDIR\opt\libexec\aorp\modules to AORP_MODULE_PATH environment variable.

3

FIRST APPLICATION

3.1 loading the modules

First of all, it is necessary to load two basic modules to activate **AutoSDK**:

- **vpw** (basic functionality);
- **vpwi-[country]** (description of LPs issued in specific country).

```
static bo_status_t _t_load_requireds(struct aorp_error *anErrPtr)
{
    static char const *_s_requireds[] = { "vpw", "vpwi-co", NULL };

    bo_status_t status;
    char const *name, **rqp;

      for (rqp = _s_requireds; NULL != (name = *rqp); ++rqp) {
        status = AorpMldLoad(name, NULL, 0, 0, anErrPtr);
        if (BoS_FAILURE(status)) {
            return (status);
            /* NOTREACHED */
        }
    }

    return (BoS_NORMAL);
}
```

> Visit the following section for more information:
>
> - aorp_error

3.2 creating the principal

To start LPR, it is necessary to create/open a principal instance. When created/opened, you can specify the parameters of its operation.

```
int main(int argc, char *argv[])
{
```

```
    aorp_object_t principal;
    struct aorp_error *err;
    ...
    principal = VpwprincOpen(0, NULL, NULL, err);
    if (NULL == principal) {
        goto L_error;
        /* NOTREACHED */
    }
    ...
    (void)AorpRelease(principal, 0, NULL); /* releasing the principal */
    ...
}
```

To explicitly specify that you want to use the license key providing the maximum available video processing speed, specify the 0 value for the **aFpsMax** parameter when opening the principal.

Visit the following sections for more information:

- VpwprincOpen
- AorpRelease
- aorp_error

3.3 configuring the principal

Once created, the principal can be set up for work. In the following example, recognition of Colombian LPs is specified.

```
static bo_status_t
_t_princ_configure(aorp_object_t aPrincipal, struct aorp_error *anErrPtr)
{
    bo_status_t status;
    struct vodi_vpw_country_id tmpl = {
        /* .code = */170/* co */,
        /* .id = */0,
        /* .sub_id = */0
    };

    assert(NULL != aPrincipal);

    status = VodiprincSetparam(aPrincipal, anErrPtr,
            VodiCTL_VPW_PLATE_TEMPLATE, &tmpl, 1);
    if (BoS_FAILURE(status)) {
        return (status);
        /* NOTREACHED */
    }
```

```
19
20      /* other parameters */
21
22      status = VodiprincApplyparam(result, anErrPtr);
23      return (status);
24  }
```

Visit the following sections for more information:

- VodiprincSetparam
- VodiprincApplyparam
- vodi_vpw_country_id
- aorp_error

3.4 recognizing the license plates

To start the recognition process, the image file should be packed into **vodi_image** structure and passed into **VodiprincProcess** operation. Each row of the image must be aligned to 4 bytes[1].
Additional options, such as sequential number of the frame, its timestamp and size, are contained in **vodi_vpw_options** structure. This structure is used when recognizing the frame-sets.
If license plates are detected, the **VodiprincProcess** operation will return an ensemble of results.

```
1   static bo_status_t
2   _t_princ_recognize(
3       aorp_object_t aPrincipal,
4       struct vodi_image *anImage,
5       size_t anAnalizeZonec,
6       vodi_rect_t const anAnalizeZonev[],
7       struct vodi_ucontext *anUserCtx,
8       aorp_object_t anInputEnsemble,
9       aorp_object_t *anOutputEnsemblePtr,
10      struct aorp_error *anErrPtr
11      )
12  {
13      bo_status_t status;
14      struct vodi_vpw_options options;
15
16      options.magic = VodiK_VPW_OPTIONS_WORK_MAGIC;
17      options.img_seqnum = 0;
18      options.img_timestamp = 0;
19      options.imgsz.sz_width = anImage->img_width;
20      options.imgsz.sz_height = anImage->img_height;
```

[1] See the VODI_IMAGE_WStride macro function (_opt\include\Vodi\Types.h). The macro function explicitly determines that the step on image lines is equal to image width aligned to 4 bytes.

```

    status = VodiprincProcess(aPrincipal, anImage, anAnalizeZonec,
        anAnalizeZonev, anUserCtx, anInputEnsemble, anOutputEnsemblePtr,
        &options, anErrPtr);

    return (status);
}
```

Visit the following sections for more information:

- VodiprincProcess
- vodi_vpw_options
- vodi_image
- aorp_error

3.5 processing the recognition results

Results processing begins with reading an ensemble of results.

```
static bo_status_t
_t_ensemble_handle(aorp_object_t anEnsemble, struct aorp_error *anErrPtr)
{
    bo_status_t status;
    size_t resc;
    aorp_object_t resv[256], *resp;
    bo_index_t from;
    _Bool overflow;

    assert(NULL != anEnsemble);

    from = 0;
    do {
        status = VodiensGet(
            anEnsemble, from, -1, __myc_namembs(resv), resv, anErrPtr
        );
        if (!BoS_DOBRE(status))
            break;

        resc = (size_t)status;
        overflow = false;
        if (__myc_namembs(resv) < resc) {
            resc = __myc_namembs(resv);
            from += (bo_index_t)resc;
            overflow = true;
```

```
        }

        resp = resv;
        do {
            status = _t_result_handle(*resp++, anErrPtr);
            if (BoS_FAILURE(status)) {
                return (status);
                /* NOTREACHED */
            }
        }
        while (--resc);
    }
    while (overflow);

    return (BoS_NORMAL);
}
```

For each ensemble result the **_t_result_handle** function is to be applied. The function uses **VodiresFetchinfo** operation to read the result data into **vodi_vpw_result_info** structure. After reading the data, the result must be destroyed with help of **VodiResultInfoDestroy** function.

```
static bo_status_t
_t_result_handle(aorp_object_t aResult, struct aorp_error *anErrPtr)
{
    bo_status_t status;
    struct vodi_result_info *info;
    struct vodi_vpw_result_info plateinfo;

    assert(NULL != aResult);

    info = (struct vodi_result_info *)&plateinfo;
    VODI_RESULT_INFO_TYPE(info) = VodiK_VPW_RESULT_INFO;
    status = VodiresFetchinfo(aResult, info, anErrPtr);
    if (BoS_FAILURE(status)) {
        return (status);
        /* NOTREACHED */
    }

    status = _t_plateinfo_handle(&plateinfo, anErrPtr);

    VodiResultInfoDestroy(info);

    return (status);
}
```

The **_t_plateinfo_handle** function can be applied on the read data in order to make use of it (for example, print the LPs text).

```
static bo_status_t
_t_plateinfo_handle(
    struct vodi_vpw_result_info *anInfo,
    struct aorp_error *anErrPtr
    )
{
    struct vodi_plate_info_spec *pis;
    struct vodi_plate *plate;

    assert(NULL != anInfo);

    pis = VODI_RESULT_INFO_SPEC(anInfo, plate);
    plate = &pis->pis_plate_variantv[0];

    /* do something */

    return (BoS_NORMAL);
}
```

Visit the following sections for more information:

- VodiensGet
- VodiresFetchinfo
- VodiResultInfoDestroy
- vodi_plate
- vodi_plate_info_spec
- aorp_error

4

DESCRIPTION OF ADDITIONAL CLI UTILITIES

4.1 description of the vpwfetch utility

4.1.1 Brief overview

The **vpwfetch** CLI application allows recognizing vehicle number plates on the images.
The application executes the following:

1. prints *header-line* into *header-file*.
2. (for each image):
   (a) prints *prefix-line* into *prefix-file*;
   (b) prints *output-line* into *output-file* (for each recognition result from the current image);
   (c) prints *sufix-line* into *sufix-file*.
3. prints *footer-line* into *footer-file*.

Commands:

1. vpwfetch -h
2. vpwfetch [parameters] --config *configuration-file*
3. vpwfetch [parameters] *input-file* [...]
4. vpwfetch [parameters] --if *input-file*
5. vpwfetch [parameters] --input-file *input-file*
6. vpwfetch [parameters] --of *output-file input-file*
7. vpwfetch [parameters] --output-file *output-file input-file*
8. vpwfetch [parameters] -S [*module-name*[, *module-name* ...]] *input-file*
9. vpwfetch [parameters] --requireds [*module-name*[, *module-name* ...]] *input-file*
10. vpwfetch [parameters] -e '*country*[*country* ...]' *input-file*
11. vpwfetch [parameters] --templates '*country*[*country* ...]' *input-file*

4.1.2 Expression semantics

Each parameter's value is interpreted as an expression. Before applying the parameter, the **vpwfetch** utility converts it into a string (or into the form where further reduction is impossible).

Table 2: Expression interpretation

Expression	Interpretation
[[empty-expression]] => empty-string	Empty expression is interpreted as an empty string.
[["""string"""]] => string	String enclosed within the three pair of double quotation marks is interpreted as is.
[['''string''']] => string	String enclosed within the three pair of single quotation marks is interpreted as is.
[["string"]] => unescape(string)	String enclosed in one pair of double quotation marks is interpreted as a string with escape sequences.
[['string']] => unescape(string)	String enclosed in one pair of single quotation marks is interpreted as a string with escape sequences.
[[word]] => word	Word (a sequence of letters and numbers) is interpreted as a string.
[[(expression)]] => [[expression]]	Expression enclosed in parenthesize is interpreted as is.
[[expression1 expression2]] => concat([[expression1]], [[expression2]])	A pair of expressions are concatenated into a string.
[[$expression]] => lookup([[expression]])	After expression processing, the value is searched (among the parameters or in process environment). If the corresponding parameter is not found, an empty string will be returned.
[[expression1 == expression2]] => strequ([[expression1]], [[expression2]])	Expressions are processed and then their strings are compared. If those strings are equal, the word 'true' will be returned (or 'false' otherwise).
[[expression1 != expression2]] => strneq([[expression1]], [[expression2]])	Expressions are processed and then their strings are compared. If those strings are equal, the word 'false' will be returned (or 'true' otherwise).
[[expression1 && expression2]] => and([[expression1]], [[expression2]])	The expression1 is to be processed. If the 'true' value is gotten, the resulting value of expression2 will be returned (or 'false' otherwise).
[[expression1 || expression2]] => or([[expression1]], [[expression2]])	The expression1 is to be processed. If the 'false' value is gotten, the resulting value of expression2 will be returned (or 'true' otherwise).
[[!expression]] => not([[expression]])	After expression processing, if 'true' word is gotten, 'false' word will be returned (or 'false' otherwise).

Expression	Interpretation
[[cond(expression1 [, ...])]]	The expression1 is to be processed. If 'true' value is gotten, the expression2 will be returned. Otherwise it all be repeated for expression3, expression4 and so on. If all the pairs were chacked and the last expression resulted without a pair, this expression will be returned (or an empty string otherwise).

Table 3: Application variables

Variable	Description
${src-path}	Path to the current image.
${src-dirname}	Path to the directory of the current image.
${src-basename}	Full name of the current image.
${src-ext}	Filename extension of the current image.
${src-name}	Filename of the current image (without the extension).
${src-seqnum}	Sequantial number of the current image.
${src-timestamp}	Timestamp of the current image or timestamp of the frame in a sequence (when processing the video)[1].
${plate-string}	The current result of vehicle plate recognition (type — string).
${plate-type}	String with the current recognition result mask.
${plate-id}	Identifier of the license plate template which corresponds to the current recognition result.
${plate-country-id}	Identifier of the license plate issuer-state (in accordance with ISO 3166).
${plate-validity}	Estimation of the current recognition result correctness.
${plate-variant-count}	Number of recognition variants for the current license plate.
${plate-outer-left}	Left coordinate of outer rectangle of the current license plate.
${plate-outer-top}	Uppermost coordinate of outer rectangle of the current license plate.
${plate-outer-right}	Right coordinate of outer rectangle of the current license plate.
${plate-outer-bottom}	Lowermost coordinate of outer rectangle of the current license plate.
${plate-exact-left}	Left coordinate of created rectangle of the current license plate.
${plate-exact-top}	Uppermost coordinate of created rectangle of the current license plate.

[1] If images are fed as input files, **vpwfetch** will sequantially process them in the interval of 40 ms (by default)

Variable	Description
${plate-exact-right}	Right coordinate of created rectangle of the current license plate.
${plate-exact-bottom}	Lowermost coordinate of created rectangle of the current license plate.
${plate-symrect-list}	List of symbol coordinates (on the current license plate image).
${plate-inversed}	Flag for the current license plate being inversed.
${plate-direction}	Vehicle movement direction. The variable can be used only when the "Dynamics" mode is activated.
${plate-angle}	Angle of the current license plate.
${plate-speed}	Vehicle speed.
${plate-index}	Identifier of the vehicle movement trajectory.
${plate-flags}	Flags for the current recognition result (this variable reflects the value of the pis_flags field which belongs to the recognition result structure).
${plate-duplicate}	Flag for a duplicated recognition result.
${plate-lost}	Flag for a vehicle movement trajectory being completed (no more data on the license plate is received). The variable can be used only when the "Dynamics" mode is activated.
${plate-invalid}	Flag for a vehicle movement trajectory being non-valid. The variable can be used only when the "Dynamics" mode is activated.
${plate-best-frame-timestamp}	Timestamp of the best image of the current license plate (within the time interval between its appearance and the current moment).
${plate-create-timestamp}	Timestamp of the vehicle appearance on the frame (first recognition result).
${plate-lost-timestamp}	Timestamp of the last frame with the vehicle on it (last recognition result).
${plate-seqnum}	Sequential number of the best license plate image (within the time interval between vehicle appearance and the current moment).
${@}	The file where the results are currently written.
${fps}	The current image processing speed.
${avgfps}	Average image processing speed (for the whole period of recognition).
${process-duration}	Time interval (in microseconds) of the current image analysis.

4.1.3 Configuration

Table 4: Input data

--encoding, --input-encoding *encoding*	Encoding of the passed parameters (UTF-8 by default).
--if, --input-file *file*	Path to the file or files with license plates to be recognized. For example: • --if */example.bmp* — path to the image; • --if *"/*.bmp"* — path to several images with the .bmp extension; • --if */example.avi* — path to the video; • --if *"/*.avi"* — path to several videos with the .avi extension.
-u, --raw [*image width*, *image height*[, *orientation*]]	Flag for the input images being in RAW format. Image orientation: TL (Top Left) or BL (Bottom Left). BL orientation is considered by default.
-S, --requireds *name* | [*name1*, ..., *nameN*] (N >= 0)	The list of AORP modules to load. Modules are to be loaded in the stated sequance.
--sp, --search-path *path* | [*path1*, ..., *pathN*] (N >= 0)	The list of AORP modules search pathes.

Table 5: Output data

--output-encoding *encoding*	Required output file encoding (UTF-8 by default). If **vpwfetch** is used on Windows OS, the console is considered as an output file (thus the required encoding is CP866 or CP1251). Expression: cond(isatty(${@}), "CP866", "CP1251").
--hf, --header-file *file*	Path to the header file. Before image analysis, value of *header-line* parameter (string) will be written to this file. It is considered by default that one file is used as a header file and output file simultaneously.
--hl, --header-line *string*	String that will be written into the header file (by default, empty string).
--pf, --prefix-file *file*	Path to the prefix file. Before image analysis, value of *prefix-line* parameter (string) will be written to this file. It is considered by default that one file is used as a prefix file and output file simultaneously.

--pl, --prefix-line *string*	String that will be written into the prefix file. By default, the following string will be written: ${src-path}"(${src-seqnum}, "${fps}"):\n".
--of, --output-file *file*	Path to the output file file. For each image analysis result, the value of *output-line* parameter (string) will be written to this file. It is considered by default that stdout will be used as an output file.
--ol, --output-line *string*	String that will be written into the output file. By default, the following string will be written: ${"plate-string"}"\n".
--sf, --sufix-file *file*	Path to the sufix file. Before image analysis, value of *sufix-line* parameter (string) will be written to this file. It is considered by default that one file is used as a sufix file and output file simultaneously.
--sl, --sufix-line *string*	String that will be written into the sufix file (by default, empty string).
--ff, --footer-file *file*	Path to the footer file. Before image analysis, value of *footer-line* parameter (string) will be written to this file. It is considered by default that one file is used as a footer file and output file simultaneously.
--fl, --footer-line string	String that will be written into the footer file (by default, empty string).

Table 6: License plate detector

-M, --plate-size-max *number*	Maximum license plate size on the frame (in pixels). By default, 200.
-m, --plate-size-min *number*	Minimal license plate size on the frame (in pixels). By default, 45.
-t, --threshold *number*	Basic image binarization level. By default, 21.
-X, --image-scale-factor [*X-factor*, *Y-factor*]	Parameters for image scaling (before license plate detection). Not specified by default.
-Z, --image-size [*width*, *height*]	Image resizing parameters (before license plate detection). Not specified by default.
-c, --plate-filter-charc *number*	Minimal number of symbols found on the license plate (with help of the basic algorithm) to continue the image analysis. By default, 0.

Table 7: Segmentator

-b, --blur *number*	Mask size used by the blur binarization algorithm. By default, 13.

Table 8: Dynamics

-y, --with-dynamic *boolean value*	Enables/disables the "Dynamics" mode (which is active by default).
-L, --comparable-time-max *time interval*	Time interval of comparing characteristics of two number plates in order to determine if they belong to the same vehicle. By default, 80 milliseconds.
-f, --comparable-symbols-min *number*	Minimal percentage of conformity between two number plates to consider them belonging to the same vehicle. By default, 80%.
--output-timeout *time interval*	Time interval from the beginning of vehicle tracking to the giving out of the first recognition result. By default, 2 seconds.
--with-duplicate *boolean value*	Enables/disables the mode of periodical giving out of recognition results (which is not active by default).
--output-period *time interval*	Time interval of recurrent giving out of vehicle movement analysis results. This parameter can only be used when the parameter --with-dublicate is enabled. By default, 500 milliseconds.
--duration-without-access *time interval*	Time period from the end of vehicle movement tracking till the last result giving out. When this time period is exceeded, the vehicle is considered gone out of the frame and is no more tracked. By default, 3 seconds.

Table 9: General settings

--config *file*	Configuration file to load.
-k, --thread-max *number*	Maximum number of parallel video analysis threads. By default, 1.
-s, --plate-star-max *number*	Maximum number of unrecognized symbols on the license plate when it is allowed to give out the result. By default, 0.
-p, --plate-validity-min *number*	Minimal estimated recognition result correctness when the result can be given out. By default, 0.

-e, --templates "*country* [*country* ...]"	Specifies the list of LP types that will be used while image analysis. By default, *. Use the lsvpwi CLI application to print information on available license plate templates (see Description of the lsvpwi utility), as well as the lsvpwc application to find out which of the templates are available according to local license keys (see Description of the lsvpwc utility). Also, get acquainted with our special grammar for represent template sets (see Return values). The grammar is used by lsvpwi and lsvpwc.
-h	Print the report and close the application.

4.1.4 Examples of commands

In case of success, the program returns the 0 code. Otherwise, the program returns the 1 code and passes the error message to strerr.

Table 10: Examples of **vpwfetch** usage

vpwfetch -S "vpwi-co" foo.bmp	Read the Colombian license plates from the foo.bmp file.
vpwfetch -S "vpwi-co" *.bmp	Read the Colombian license plate numbers from series of *.bmp files.
vpwfetch -S "vpwi-co" -y 1 *.bmp	Read the Colombian license plates from series of *.bmp files in the "Dynamics" mode.

4.2 description of the lsvpwi utility

4.2.1 Overview

The **lsvpwi** CLI application prints information on license plate templates which are supported by the used **AutoSDK** version.

4.2.2 Options and examples

> Before using the utility, it is recommended to create the AORP_MODULE_PATH environment variable, which value would be the path to the directory with the country modules (vpwi-[country_code].dll). In relation to the directory where the lsvpwi utility is located, the country modules are located in ..\libexec\aorp\modules.

If the environment variable is not created, you have to explicitly specify the path to country modules using the -m or --m options for each application call.

Table 11: lsvpwi options

Option	Meaning
--help	Print the list of lsvpwi commands and their meanings.
-m --modules-path --mp FilePath	Specify the directory path where country modules are located.
-h --human-readable --hr	Print the literal country codes (according to ISO 3166) as part of LP template identifiers, not the numerical ones. For example, when using this option, instead of the 170.[template_number] identifier the co.[template_number] will be output (i.e. Colombian LP template). This option makes it easier to identify to which country LP templates belong.
-r --react	Print error notifications (for example, when templates of the requested country are not supported).
-f --force	Ignore runtime error notifications (the option is enabled by default).
-o --output-file --of FilePath --stdout	Output information on requested templates to stdout (by default) or write it into the file.

-s --output-fields --fields Fields	Specify the informational field names to be shown for requested template group (only the standard fields are output by default). When inputting the required fields, sepatate them by comma. Available fields: • id — new LP template identifier. • oldid — deprecated LP template identifier. • mask — LP template mask (Z — letter, X — number). • linec — number of rows on the LP represented by the template. • symc — number of symbols on the LP represented by the template. • print_format — format in which the recognized license number (represented by the template) will be printed (as a string), i.e. the symbols' output order. [2] • real_size — LP size, mm (according to national standards). • background, foreground — LP background color and symbols color respectively.[3] • properties — reserved.
-t --output-format --format Format	Select the output information formatting: 'table' (by default), 'json' or 'cut' (CSV).

If you do not specify one or more country codes or names which templates information is to be printed, you will get information on *all* the available templates.

Table 12: lsvpwi command examples

lsvpwi nl de fr --hr -s id,oldid	Show identifiers (new and deprecated ones) for LP templates of the Netherlands, Germany and France. The identifiers should contain literal country codes, not numerical ones.
lsvpwi at -s "id, print_format, linec, symc"	Show the print format, number of rows & symbols for each Austrian LP template.

[2] The output format is based on position variables like $n, where n is the sequence number of the symbol. Printing format is usually specified by the LP template (usually from left to right and from up to buttom). If the parameter has no value, the corresponding license number will be printed as is. Also, this parameter can specify the symbols that are not to be printed or are to be ignored during the recognition (e.g. en-dash).

[3] The constants are defined in <Vodi/clrsType.h>.

lsvpwi Colombia --output-file Colombian_templates.txt --output-format json	Write information on Colombian LP templates into the Colombian_templates.txt file (in JSON format).

4.3 description of the lsvpwc utility

4.3.1 Overview

The **lsvpwc** CLI application scans the local machine for available Sentinel HASP license keys, and, in case of success, prints information on them.

4.3.2 Options

> Before using the utility, it is recommended to create the AORP_MODULE_PATH environment variable, which value would be the path to the directory with the country modules (vpwi-[country_code].dll). In relation to the directory where the lsvpwc utility is located, the country modules are located in ..\libexec\aorp\modules.

If the environment variable is not created, you have to explicitly specify the path to country modules using the -m or --m options for each application call (see the table below).

Table 13: lsvpwc options

Option	**Meaning**
--help	Print the list of lsvpwc commands and their meanings.
-m --modules-path --mp FilePath	Specify the directory path where country modules are located.
-o --output-file --of FilePath --stdout	Output information on available licenses to stdout (by default) or write it into the file.
-t --output-format --format Format	Select the output information formatting: 'table' (by default), 'json' or 'cut' (CSV).
-g --group	Summarize and output all the capabilities, enabled by all the license keys available on the current machine.

Example of application's output:

```
$lsvpwc
+----------+----------+-------+------------+-----------+
|    key_id|key_series|fps_max|nthreads_max|  templates|
+----------+----------+-------+------------+-----------+
|1160193448|    107392|     25|           4|ru kz uz kg|
|1528129579|    106763|     25|          16|   ua ru su|
+----------+----------+-------+------------+-----------+

$lsvpwc -g
+------+----------+-------+------------+-----------------+
|key_id|key_series|fps_max|nthreads_max|        templates|
+------+----------+-------+------------+-----------------+
|  ----|      ----|     25|          20|kz kg ru ua su uz|
+------+----------+-------+------------+-----------------+
```

Licensing parameters:

- key_id — unique license key identifier.
- key_series — unique identifier of **AutoSDK** vendor (in terms of Sentinel HASP, *batch code*).[4]
- fps_max — maximum number of fps which can be processed by **AutoSDK** enabled by the license key.
- templates — list of countries, which license plates can be recognized by used **AutoSDK** instance.
- flags — additional license key parameters:
  - D — (output-delay) the flag is printed for those license keys which cause 30-second delays while giving out the recognition results.

[4] Seenaptecidentifiers: EAOWT and AAOTB (numerical equivalents are 107392 and 106763 respectively.

5

DESCRIPTION OF OPERATIONS WITH AORP OBJECTS

5.1 vpwprincopen

5.1.1 Name

VpwprincOpen — opens/creates principal.

5.1.2 Syntax

```
#include <Vodi/objects/Vpwprinc.h>

aorp_object_t
__attribute__((nonnull(1,4))
VpwprincOpen(
    aorp_opflags_t Flags /* = 0 */,
    bo_pointer_t aMemory /* = NULL */,
    struct vodi_vpw_princ_param *aParm,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.1.3 Description

- The **VpwprincOpen** function creates the principal and returns pointer to it.
- The **Flags** argument allows managing the object construction. 0 is passed by default.
- The **aMemory** argument is used to pass a pointer to the memory allocated for object creation. NULL is passed by default (in this case, the memory will be allocated by the constructor).
- The **aParm** argument is used for passing the initial parameters of principal operation. For detailed information, see the VodiprincSetparam section.

Table 14: Fields of the **vodi_vpw_princ_param** structure

Field	Description
enable	Equivalent of VodiCTL_VPW_PRINCIPAL_ENABLE.
thread_max	Maximum number of recognition threads that, if necessary, are created while principal operation. If the value is 0, the recognition process will occur in the same thread that starts it. Setting the value greater than number of CPU cores with consideration of Hyper Threading is not recommended.
image_width	Equivalent of VodiCTL_VPW_IMAGE_WIDTH.
image_height	Equivalent of VodiCTL_VPW_IMAGE_HEIGHT.
image_brightnes	Equivalent of VodiCTL_VPW_IMAGE_BRIGHTNES.
image_contrast	Equivalent of VodiCTL_VPW_IMAGE_CONTRAST.
image_blur	Equivalent of VodiCTL_VPW_IMAGE_BLUR.
image_treshold	Equivalent of VodiCTL_VPW_IMAGE_THRESHOLD.
image_reversed	Deprecated. Should be set to 0.
image_angle	Equivalent of VodiCTL_VPW_IMAGE_ANGLE.
analysed_zone	Equivalent of VodiCTL_VPW_IMAGE_ANALYSE_ZONE.
analysed_zone_count	Current number of elements in the analysed_zone array.
plate_size_max	Equivalent of VodiCTL_VPW_PLATE_SIZE_MAX.
plate_size_min	Equivalent of VodiCTL_VPW_PLATE_SIZE_MIN.
plate_inverse_analyse	Equivalent of VodiCTL_VPW_PLATE_INVERSE_ANALYSE.
plate_probability_min	Equivalent of VodiCTL_VPW_PLATE_PROBABILITY_MIN.
plate_star_max	Equivalent of VodiCTL_VPW_PLATE_STAR_MAX.
md_enable	Deprecated. Should be set to 0.
md_cell_size	Deprecated. Should be set to 0.
md_threshold	Deprecated. Should be set to 0.
md_square_min	Deprecated. Should be set to 0.
md_mask	Deprecated. Should be set to 0.
md_mask_h_size	Deprecated. Should be set to 0.
md_mask_v_size	Deprecated. Should be set to 0.
dynamic_enable	Equivalent of VodiCTL_VPW_DYNAMIC_ENABLE.
log_settings	Equivalent of VodiCTL_VPW_LOG_SETTINGS

5.1.4 Return values

In case of success, the **VpwprincOpen** function returns a pointer to the created object. Otherwise, the function will return NULL as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.2 aorpretain

5.2.1 Name

AorpRetain — retains an AORP object.

5.2.2 Syntax

```
#include <Bo/services/Ucntl.h>

ssize_t
__attribute__((nonnull(1))
AorpRetain(
aorp_object_t aThis,
struct aorp_error *anErrPtr /* = NULL */
);
```

5.2.3 Description

The **AorpRetain** function retains the AORP object passed by the **aThis** argument. As a result, the function returns the number of references to the object.

5.2.4 Return values

In case of success, the **AorpRetain** function returns value greater or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.3 aorprelease

5.3.1 Name

AorpRelease — releases an AORP object.

5.3.2 Syntax

```
#include <Bo/services/Ucntl.h>

ssize_t
__attribute__((nonnull(1))
AorpRelease(
    aorp_object_t aThis,
    aorp_opflags_t Flags /* = 0 */,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.3.3 Description

The **AorpRelease** function releases the AORP object passed by the **aThis** argument.

5.3.4 Return values

In case of success, the **AorpRelease** function returns the remaining number of references to the object. Return value 0 means that this function's call destroyed the object. The function also returns negative value as an error status and fills the structure specified by the **anErrPtr** argument with an error description.

5.4 vodiprincgetparam

5.4.1 Name

VodiprincGetparam — gets the parameter value.

5.4.2 Syntax

```
#include <Vodi/services/Vodiprinc.h>

bo_status_t
__attribute__((nonnull(1))
VodiprincGetparam(
    aorp_object_t aThis,
    struct aorp_error *anErrPtr /* = NULL */,
    int aParam,
    ...
    );
```

5.4.3 Description

The **VodiprincGetparam** function returns a value of the parameter specified by the **aParam** argument. Depending on the parameter type, additional pointers can be passed to the function to store the current parameter values. For detailed information on available parameters, see the VodiprincSetparam section.

5.4.4 Return values

In case of success, the **VodiprincGetparam** function returns the current parameter value that is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.5 vodiprincsetparam

5.5.1 Name

VodiprincSetparam — sets the parameter.

5.5.2 Syntax

```
#include <Vodi/services/Vodiprinc.h>

bo_status_t
__attribute__((nonnull(1))
VodiprincSetparam(
    aorp_object_t aThis,
    struct aorp_error *anErrPtr /* = NULL */,
    int aParam,
    ...
    );
```

5.5.3 Description

The **VodiprincSetparam** function sets the value of the parameter specified by the **aParam** parameter. Depending on parameter type, additional arguments for setting values can be passed to the function.

Table 15: VodiprincSetparam parameters

Parameter	Description
VodiCTL_VPW_PRINCIPAL_ENABLE	Deprecated/unsupported.
VodiCTL_VPW_PLATE_INVERSE_ANALYSE	Deprecated / unsupported.
VodiCTL_VPW_PLATE_PROBABILITY_MIN	Minimal percentage of conformity between the LPR result and the corresponding LP template for the result to be considered as a license plate number. With help of this parameter the results filtering by validity is performed. Type — unsigned.
VodiCTL_VPW_PLATE_STAR_MAX	Maximum number of unrecognized characters (on the LP) to still consider the obtained result as a license plate number. Type — unsigned.
VodiCTL_VPW_PLATE_SIZE_MAX	Maximum LP width (in pixels) to still consider the obtained result as a license plate number. Type — unsigned.
VodiCTL_VPW_PLATE_SIZE_MIN	Minimal LP width (in pixels) to still consider the obtained result as a license plate number. Type — unsigned.

Parameter	Description
VodiCTL_VPW_PLATE_EXTRA_ANGLE_ANALYSE	Enable/disable the algorithm of processing the LP image perspective. Type — int(boolean).
VodiCTL_VPW_PLATE_EXTRA_RANGES_ANALYSE	Enable/disable more accurate finding candidates of plates. Type — int(boolean).
VodiCTL_VPW_PLATE_FILTER_ROFACTOR	Coefficient of filtering results by so-called image density (the ratio of the white pixels number to the number of all pixels). First strategy. Type — unsigned. The coefficient is used while image binarization and its optimal values are determined by the **AutoSDK** developers with help of their own test materials. This is a service parameter and its value should be set with consideration of recommendations provided by **AutoSDK** technical support specialists.
VodiCTL_VPW_ PLATE_FILTER_RODROPFACTOR	Coefficient of filtering results by so-called image density (the ratio of the white pixels number to the number of all pixels). Second strategy. Type — unsigned. The coefficient is used while image binarization and its optimal values are determined by the **AutoSDK** developers with help of their own test materials. This is a service parameter and its value should be set with consideration of recommendations provided by **AutoSDK** technical support specialists.
VodiCTL_VPW_ PLATE_IMAGE_SCALE_FACTOR	Coefficients of image resize that is performed before LP detection on the frame. Resizing the image can reduce the time required for LP search. This parameter values are absolute and act as two resizing coefficients (for image height and width). This parameter and the VodiCTL_VPW_PLATE_IMAGE_SIZE parameter are mutually exclusive. Type — **struct vodi_scale_factor** *. Usage example: if it is needed to make the image width 2 times smaller and the image height 3 times smaller, the parameter value is set to {2, 3}.

Parameter	Description
VodiCTL_VPW_PLATE_IMAGE_SIZE	Size the image must be reduced to. After resizing, the LP detection is performed on the image. Resizing the image can reduce the time required for LP search. This parameter and the VodiCTL_VPW_PLATE_IMAGE_SCALE_FACTOR are mutually exclusive. Type — **struct vodi_size** *. This parameter values are absolute.
VodiCTL_VPW_PLATE_TEMPLATE	Enable/disable the template defined by the first argument. The second argument defines what is to be done (0 — disable, !0 — enable). Type of the first argument — **struct vodi_vpw_country_id** *. Type of the second argument — int(boolean).
VodiCTL_VPW_PLATE_FILTER_SYMCOUNT	Enable/disable the basic algorithm of filtering LPs by minimal number of symbols found on them. If the algorithm is enabled (the value greater than 0), the initial symbols search is performed on the LP candidate (geometry, proportions). If the found symbols number is less than this parameter value, the candidate is not considered as a license plate. Type — unsigned.
VodiCTL_VPW_PLATE_PRECISE_ANALYSE	Enable/disable advanced image analysis. Improves recognition quality under extreme conditions (rain, snow, unproper LPR camera settings). When used, this parameter increases processing time by 20-30% (depending on an image size). Generally, this parameter doesn't have an effect when recognizing LPs under normal conditions. Type — int(boolean).
VodiCTL_VPW_DNN_DEVICES	A list of devices on which the Deep Neural Network will run. Type of the first argument — **vodi_devtyp_t** *. Type of the second argument — unsigned.
VodiCTL_VPW_PLATECANDS_METHODS	A set of methods for obtaining plate candidates. Available methods are VodiF_VPW_PLATECANDS_BY_MORPH and VodiF_VPW_PLATECANDS_BY_DNN. Type — unsigned.

Parameter	Description
VodiCTL_VPW_ANALYSE_LEVEL	The plate analysing level. Available levels are VodiK_VPW_PLATECANDS_ANALYSE, VodiK_VPW_SYMCANDS_ANALYSE and VodiK_VPW_TEXT_ANALYSE. Type — int.
VodiCTL_VPW_IMAGE_WIDTH	Deprecated / unsupported.
VodiCTL_VPW_IMAGE_HEIGHT	Deprecated / unsupported.
VodiCTL_VPW_IMAGE_BRIGHTNES	Deprecated / unsupported.
VodiCTL_VPW_IMAGE_CONTRAST	Deprecated / unsupported.
VodiCTL_VPW_IMAGE_BLUR	Service parameter (for internal use). The recommended value is 13. Type — int.
VodiCTL_VPW_IMAGE_THRESHOLD	The binarization threshold of conversion of color pixels into black& white ones. The recommended value is 21. Type — unsigned.
VodiCTL_VPW_IMAGE_ANALYSE_ZONE	Array of recognition zones where LPs are considered to appear. Type of the first argument — **struct vodi_rect** *. Type of the second argument — unsigned.
VodiCTL_VPW_IMAGE_AZONE_MASK	Mask image of the analized zone.Type — **struct vodi_image** *.
VodiCTL_VPW_IMAGE_NOT_ANALYSE_ZONE	Deprecated/unsupported.
VodiCTL_VPW_IMAGE_ANGLE	Deprecated/unsupported.
VodiCTL_VPW_DYNAMIC_ENABLE	Enable/disable the Dynamics mode. Type — int(boolean).
VodiCTL_VPW_DYNAMIC_OUTPUT_TIMEOUT	Minimal duration of LP tracking (in microseconds) before the LPR result can be obtained by user. This parameter can only be used with the Dynamics mode on. In this mode, the vehicle movement is being tracked and the final result is to be released when the time corresponding to this parameter value has passed. In this case, resulting data is guaranteed to be more reliable. If this parameter value is set to 0, the user will get the very first result of the current LP recognition. When the time corresponding to this parameter value has passed, LP movement tracking continues until the vehicle leaves the frame. Type — bo_usec_t.

Parameter	Description
VodiCTL_VPW_DYNAMIC_OUTPUT_PERIOD	Time period (in microseconds) of giving out the recognition result. This parameter is to be used only if the VodiCTL_VPW_ DYNAMIC_WITH_DUPLICATE parameter is set. Type — bo_usec_t.
VodiCTL_VPW_DYNAMIC_WITH_DUPLICATE	Enable/disable periodical giving out of recognition results. Type — int(boolean).
VodiCTL_VPW_ DYNAMIC_DURATION_WITHOUT_ACCESS	Maximum allowed time of LP absence on the frame (in microseconds). If time corresponding to this parameters value has passed and the LP didn't reappear, the LP is no more considered overlapped (it is considered lost). The LPR result is obtained by user with the VodiF_RESULT_LOST flag. Type — bo_usec_t.
VodiCTL_VPW_LOG_SETTINGS	Enable/disable the logging of all recognition parameters. Type — int(boolean).

5.5.4 Return values

In case of success, the **VodiprincSetparam** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.6 vodiprincapplyparam

5.6.1 Name

VodiprincApplyparam — applies the parameters.

5.6.2 Syntax

```
#include <Vodi/services/Vodiprinc.h>

bo_status_t
__attribute__((nonnull(1))
VodiprincApplyparam(
    aorp_object_t aThis,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.6.3 Description

The **VodiprincApplyparam** function applies the parameters set by the **VodiprincSetparam** function.

5.6.4 Return values

In case of success, the **VodiprincApplyparam** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.7 vodiprinccontrol

5.7.1 Name

VodiprincControl — controls the principal.

5.7.2 Syntax

```
# include <Vodi/services/Vodiprinc.h>

bo_status_t
__attribute__((nonnull(1)))
VodiprincControl(
    aorp_object_t aThis,
    struct aorp_error *anErrPtr,
    int aCommand,
    ...
    );
```

5.7.3 Description

The **VodiprincControl** function receives the following parameters:

- **aThis** — the principal instance.
- **anErrPtr** — pointer to the structure which describes an error.
- **aCommand** — command to execute.
- **...** — specific command's arguments.

Used commands:

- **VodiCTL_GETVPWI** — command to return the set of templates which are used at the moment. char ** parameter is expected (a pointer to the string where the present templates set will be written).
- **VodiCTL_ADDVPWI** — command to add the specified templates set to the set currently used. char const * parameter is expected (a set of templates to add).
- **VodiCTL_SETVPWI** — command to flush the set of templates currently used and substitute them with a set passed as a parameter. char const * parameter is expected.
- **VodiCTL_DELVPWI** — command to delete the templates (specified by the parameter) from the set of templates currently used. char const * is expected.

5.7.4 Return values

In case of success, the **VodiprincControl** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

Visit the following sections for more information:

- Description
- Return values

5.8 vodiprincprocess

5.8.1 Name

VodiprincProcess — analyzes the image.

5.8.2 Syntax

```
#include <Vodi/services/Vodiprinc.h>

bo_status_t
__attribute__((nonnull(1,2,7))
VodiprincProcess(
    aorp_object_t aThis,
    struct vodi_image *anImage,
    size_t anAnalizeZonec /* = 0 */,
    vodi_rect_t const anAnalizeZonev[] /* = NULL */,
    struct vodi_ucontext const *anUserCtx /* = NULL */,
    aorp_object_t anInputEnsemble /* = NULL */,
    aorp_object_t *anOutputEnsemblePtr,
    void *Options /* = NULL */,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.8.3 Description

- The **VodiprincProcess** function analyzes the image passed by the **anImage** parameter.
- You can mark specific areas (recognition zones) on the image. The function will analyze only the image fragments represented by these recognition zones. Their number is specified by the **anAnalizeZonec** argument, and the pointer to an array of prepared rectangles is specified by the **anAnalizeZonev** argument. If 0 is passed as the number of areas, the **anAnalizeZonev** argument is ignored, thus the entire image will be analyzed.
- User context can be passed by the **anUserCtx** argument. This context will be linked with every analysis result. Detailed information on user context can be found in the vodi_ucontext section.
- Each analysis result can be inserted into ensemble which was prepared by user. This ensamble is passed to the function by the **anInputEnsemble** argument. If this argument is NULL, a new ensemble will be created for output.
- Pointer to the output ensamble is represented by the **anOutputEnsemblePtr** parameter. If the **anInputEnsemble** argument is NULL, resulting function output will be pointed by the **anOutputEnsemblePtr** parameter.
- For analysis/search of license plate, an additional parameter can be passed by the **Options** argument. In this case, the pointer must hold a reference to **struct vodi_vpw_options**.

5.8.4 Return values

In case of success, the **VodiprincProcess** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.9 vodiprincflush

5.9.1 Name

VodiprincFlush — flushes inner buffers.

5.9.2 Syntax

```
#include <Vodi/services/Vodiprinc.h>

bo_status_t
__attribute__((nonnull(1,3))
VodiprincFlush(
    aorp_object_t aThis,
    aorp_object_t anInputEnsemble /* = NULL */,
    aorp_object_t *anOutputEnsemblePtr,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.9.3 Description

The **VodiprincFlush** function flushes inner buffers containing the analysis results which were obtained in the Dynamics mode.
Each result from the buffer is stored in the ensemble passed by the **anInputEnsemble** argument. If the **anInputEnsemble** parameter is NULL, a new ensemble will be created.
The results ensemble is passed by the **anOutputEnsemblePtr** output argument.

5.9.4 Return values

In case of success, the **VodiprincFlush** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.10 vodienscount

5.10.1 Name

VodiensCount — returns number of results contained in the ensemble.

5.10.2 Syntax

```
#include <Vodi/services/Vodiens.h>

ssize_t
__attribute__((nonnull(1))
VodiensCount(
    aorp_object_t aThis,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.10.3 Return values

In case of success, the **VodiensCount** function returns the number of results in the ensemble. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.11 vodiensclear

5.11.1 Name

VodiensClear — clears the ensemble.

5.11.2 Syntax

```
#include <Vodi/services/Vodiens.h>

bo_status_t
__attribute__((nonnull(1))
VodiensClear(
    aorp_object_t aThis,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.11.3 Description

The **VodiensClear** function clears the ensemble passed by the **aThis** argument. Then the **AorpRelease** function is applied to all results in the ensemble.

5.11.4 Return values

In case of success, the **VodiensClear** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.12 vodiensadd

5.12.1 Name

VodiensAdd — adds the result.

5.12.2 Syntax

```
#include <Vodi/services/Vodiens.h>

bo_status_t
__attribute__((nonnull(1,2))
VodiensAdd(
    aorp_object_t aThis,
    aorp_object_t aResult,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.12.3 Description

The **VodiensAdd** function inserts the result passed by the **aResult** argument into the ensemble of the **aThis** argument. The **AorpRetain** function is applied to the result.

5.12.4 Return values

In case of success, the **VodiensAdd** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.13 vodiensremove

5.13.1 Name

VodiensRemove — removes results from the ensamble.

5.13.2 Syntax

```
#include <Vodi/services/Vodiens.h>

bo_status_t
__attribute__((nonnull(1,3))
VodiensRemove(
    aorp_object_t aThis,
    bo_index_t anIndex,
    aorp_object_t *aResultPtr,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.13.3 Description

The index of the result that should be removed from the ensemble is specified by the **anIndex** argument (indices start from 0). If a negative value is passed, result selection will be performed from the end of the ensemble. Removed result is stored in the output argument **aResultPtr**. The **AorpRetain**/**AorpRelease** functions are not applied to the removed result.

5.13.4 Return values

In case of success, the **VodiensRemove** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.14 vodiensdelete

5.14.1 Name

VodiensDelete — deletes results from the ensamble.

5.14.2 Syntax

```
#include <Vodi/services/Vodiens.h>

bo_status_t
__attribute__((nonnull(1))
VodiensDelete(
    aorp_object_t aThis,
    bo_index_t anIndex,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.14.3 Description

The index of the result that should be deleted from the ensemble is specified by the **anIndex** argument (indices start from 0).If a negative value is passed, result selection will be performed from the end of the ensemble. The **AorpRelease** function will be applied to the selected result.

5.14.4 Return values

In case of success, the **VodiensDelete** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.15 vodiensget

5.15.1 Name

VodiensGet — gets results from the ensamble.

5.15.2 Syntax

```
#include <Vodi/services/Vodiens.h>

ssize_t
__attribute__((nonnull(1))
VodiensGet(
    aorp_object_t aThis,
    bo_index_t aFrom,
    bo_index_t aTo,
    size_t aCount,
    aorp_object_t aResultv[],
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.15.3 Description

- The **VodiensGet** function selects results from the ensemble passed by the **aThis** argument. Results are selected from the range that was specified with the [**aFrom**, **aTo**] indices. Index with negative value points to an element located at the end of the ensemble.
- The maximum number of the results that can be written into the **aResultv** buffer is specified by the **aCount** argument. If number of selected results is greater than buffer capacity, there won't be any errors.
- The **AorpRetain**/**AorpRelease** functions are not applied to the selected results.

5.15.4 Return values

In case of success, the **VodiensGet** function returns the number of selected results (which can be greater than value of the **aCount** argument). Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.16 vodiensunique

5.16.1 Name

VodiensUnique — removes non-unique results from the ensemble.

5.16.2 Syntax

```
#include <Vodi/services/Vodiens.h>

ssize_t
__attribute__((nonnull(1,2))
VodiensUnique(
    aorp_object_t aThis,
    aorp_object_t aPrincipal,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.16.3 Description

The **VodiensUnique** function leaves only the unique results in the ensemble which was passed by the **aThis** argument. The **aPrincipal** argument should point to the principal which is to perform this operation.

5.16.4 Return values

In case of success, the **VodiensUnique** function returns the number of results remaining in the ensemble. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.17 vodienscombine

5.17.1 Name

VodiensCombine — combines the results.

5.17.2 Syntax

```
#include <Vodi/services/Vodiens.h>

ssize_t
__attribute__((nonnull(1,2))
VodiensCombine(
    aorp_object_t aThis,
    aorp_object_t aPrincipal,
    int anWithMerge,
    size_t anEnsc,
    aorp_object_t anEnsv[],
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.17.3 Description

- The **VodiensCombine** function joins several ensambles into one (specified by the **aThis** argument).
- The **aPrincipal** argument should point to the principal which is to perform this operation.
- The number of ensambles to be combined and the array with them are to be defined by the **anEnsc** and **anEnsv** arguments accordingly. After joining, results will be deleted from the input ensembles.
- If the **anWithMerge** field has a non-zero value, the **VodiensCombine** function will write only unique results into the output (resulting) ensemble.

5.17.4 Return values

In case of success, the **VodiensCombine** function returns the number of results remaining in the output ensemble. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.18 vodienscombine_v2

5.18.1 Name

VodiensCombine_v2 — combines the results.

5.18.2 Syntax

```
#include <Vodi/services/Vodiens.h>

ssize_t
__attribute__((nonnull(1,2))
VodiensCombine_v2(
    aorp_object_t aThis,
    aorp_object_t aPrincipal,
    aorp_opflags_t Flags,
    size_t anEnsc,
    aorp_object_t anEnsv[],
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.18.3 Description

The **VodiensCombine_v2** function joins several ensambles into one (specified by the **aThis** argument). The **aPrincipal** argument should point to the principal which is to perform this operation. The number of ensambles to be combined and the array with them are to be defined by **anEnsc** and **anEnsv** arguments accordingly.
The following flags can be passed to the function by the **Flags** argument:

- VodiensF_COMBINE_WITH_MERGE — write only the unique results into the output (resulting) ensemble;
- VodiensF_COMBINE_WITH_DUP — don't delete results from input ensembles (passed by the **anEnsv** argument), just copy them into the resulting ensemble.

5.18.4 Return values

In case of success, the **VodiensCombine_v2** function returns the number of results remaining in the output ensemble. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.19 vodiresgetuserdata

5.19.1 Name

VodiresGetuserdata — gets the user's context.

5.19.2 Syntax

```
#include <Vodi/services/Vodires.h>

bo_status_t
__attribute__((nonnull(1,2))
VodiresGetuserdata(
    aorp_object_t aThis,
    struct vodi_ucontext *anUserCtx,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.19.3 Description

The **VodiresGetuserdata** function returns the user's context by writing it into the structure pointed by the **anUserCtx** argument. The **VODI_UCONTEXT_DRetain** operation won't be applied to the user context.

5.19.4 Return values

In case of success, the **VodiresGetuserdata** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.20 vodiressetuserdata

5.20.1 Name

VodiresSetuserdata — sets the user's context.

5.20.2 Syntax

```
#include <Vodi/services/Vodires.h>

bo_status_t
__attribute__((nonnull(1,2))
VodiresSetuserdata(
    aorp_object_t aThis,
    struct vodi_ucontext *anUserCtx,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.20.3 Description

The **VodiresSetuserdata** function sets user's context passed by the **anUserCtx** argument. The **VODI_UCONTEXT_DRetain** operation is applied to the context.

5.20.4 Return values

In case of success, the **VodiresSetuserdata** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.21 vodiresfetchinfo

5.21.1 Name

VodiresFetchinfo — reads the information.

5.21.2 Syntax

```
#include <Vodi/services/Vodires.h>

bo_status_t
__attribute__((nonnull(1,2))
VodiresFetchinfo(
    aorp_object_t aThis,
    struct vodi_result_info *anInfo,
    struct aorp_error *anErrPtr /* = NULL */
    );
```

5.21.3 Description

The **VodiresFetchinfo** function reads the information on the recognition result into the buffer specified by the **anInfo** argument.
It is required to set the **ri_type** field of the **anInfo** argument. The following values are available:

- VodiK_VPW_RESULT_INFO — information on the number. The **anInfo** pointer must point to the **vodi_vpw_result_info** structure.
- VodiK_FCW_RESULT_INFO — reserved.

User must destroy the information which was written into the buffer (specified by the **anInfo** parameter) by calling the **VodiResultInfoDestroy** function.

5.21.4 Return values

In case of success, the **VodiresFetchinfo** function returns the value which is greater than or equal to 0. Otherwise, the function will return negative value as an error status and fill the structure specified by the **anErrPtr** argument with an error description.

5.22 vodiresultinfodestroy

5.22.1 Name

VodiResultInfoDestroy — destroys the information on the result.

5.22.2 Syntax

```
#include <Vodi/Vodilib.h>

void
__attribute__((nonnull(1))
VodiResultInfoDestroy(
    struct vodi_result_info *anInfo
    );
```

5.22.3 Description

The **VodiResultInfoDestroy** function destroys the information on the recognition result specified by the **anInfo** argument. This function is used along with the **VodiresFetchinfo** function.

6

DESCRIPTION OF STRUCTURES

6.1 vodi_vpw_country_id

6.1.1 Name

struct vodi_vpw_country_id — identifier of LP template.

6.1.2 Syntax

```
#include <Vodi/Vpwtypes.h>

struct vodi_vpw_country_id {
    unsigned int code;
    unsigned int id;
    unsigned int sub_id;
};
```

6.1.3 Description

- **code** — issuer-state code (in accordance with ISO 3166-1 numeric).
- **id** — LP template identifier. The 0 value is allowed and stands for all the templates that can be specified by the **code** field.
- **sub_id** — reserved (the value mast be set to 0).

6.2 vodi_image

6.2.1 Name

struct vodi_image — image to analyze.

6.2.2 Syntax

```
#include <Vodi/Types.h>

struct vodi_image {
    unsigned      img_flags;
    long          img_width;  /* in pixels */
    long          img_height; /* in pixels */
    unsigned      img_bpp;    /* bits per pixel */
    bo_pointer_t  img_base;
};
```

6.2.3 Description

- **img_flags** — this parameter value must be VodiF_IMAGE_PLAIN (no other flags), if the **vodi_image** object is created by user. If the object is created by function from VodiImage family (e.g. VodiImageCreate), the field will be set by that function automatically.
- **img_width** — image width.
- **img_height** — image height.
- **img_bpp** — bits per pixel. The value must be 8.
- **img_base** — pointer to the first row of the image. Each row of the image must be aligned to 4 bytes.

6.3 vodi_vpw_options

6.3.1 Name

struct vodi_vpw_options — additional analytics parameters.

6.3.2 Syntax

```
#include <Vodi/Vpwtypes.h>

struct vodi_vpw_options {
    int          magic;
    vodi_size_t  imgsz;         /* reserved; deprecated */
    u_int32_t    img_seqnum;    /* original image sequence mark */
    bo_utime_t   img_timestamp; /* original image timestamp */
};
```

6.3.3 Description

- **magic** — reserved for determining the structure type to use (only the vodi_vpw_options structure is currently available). Assign VodiK_VPW_OPTIONS_WORK_MAGIC to this parameter.
- **imgsz** — deprecated.
- **img_seqnum** — sequantial number of the image (within the frame-set). The field is used with the Dynamics mode on.
- **img_timestamp** — image timestamp (in microseconds). Used for timing analysis while license plates tracking. The field is used by the following functions (with the Dynamics mode on):
  - **VodiensUnique**;
  - **VodiensCombine** (in WithMerge mode);
  - **VodiensCombine_v2** (with the VodiensF_COMBINE_WITH_MERGE flag).

6.4 vodi_ucontext

6.4.1 Name

struct vodi_ucontext — user's context.

6.4.2 Syntax

```
#include <Vodi/Types.h>

struct vodi_ucontext {
    vodi_uctx_dup_fn       uctx_dup;
    vodi_udata_retain_fn   uctx_dretain;
    vodi_udata_release_fn  uctx_drelease;
    vodi_uctx_change_fn    uctx_change;
    bo_pointer_t           uctx_udata;
};
```

6.4.3 Description

This structure is provided to store the user's context which can be passed to the **VodiprincProcess** function. While image analysis, this context will be associated with the corresponding LPR result.

- When linking context and LPR result, the operation determined by the **uctx_dretain** field will be called for the context. If the value of the field is NULLF (NULL function pointer), no operation will be called.
- When running the **VodiprincProcess** function, need for result dublicate can occur. In this case, the operation determined by the **uctx_dup** field will be called for the context. If the value of the field is NULLF (NULL function pointer), no operation will be called.
- When the LPR result must be deleted, the operation determined by the **uctx_drelease** field will be called for the context. If the value of the field is NULLF (NULL function pointer), no operation will be called.
- When the Dynamics mode is active, LPR results obtained from several frames are to be merged. In this case, the operation determined by the **uctx_change** field is called for results contexts. If the value of the field is NULLF (NULL function pointer), no operation will be called.
- The **uctx_udata** pointer can hold the reference to additional data.

6.5 vodi_plate_info_spec

6.5.1 Name

struct vodi_plate_info_spec — recognition result specification.

6.5.2 Syntax

```
#include <Bo/aorp/Vpwtypes.h>

structvodi_plate_info_spec {
    size_t           pis_plate_variantc;
    vodi_plate_t     pis_plate_variantv[VodiL_VPW_VARIANT_MAX];

    vodi_rect_t      pis_outer_rect;
    vodi_rect_t      pis_exact_rect;
    vodi_quad64f_t   pis_exact_quad;

    _Bool            pis_inversed;
    vodi_motiondir_t pis_direction;
    double           pis_angle;
    double           pis_speed;
    size_t           pis_index;
    unsigned         pis_flags;
    _Bool            pis_move_in;
    size_t           pis_status;

    vodi_ucontext_t  pis_uctx;
    vodi_image_t     pis_image;

    bo_utime_t       pis_bftime;
    bo_utime_t       pis_ctime;
    bo_utime_t       pis_ltime;

    vodi_size_t      pis_imgsz;
    u_int32_t        pis_seqnum;
};
```

6.5.3 Description

Table 16: struct vodi_plate_info_spec parameters

Parameter	Description
pis_plate_variantc	Number of LP recognition result variants. Each variant is represented by vodi_plate type (see vodi_plate).
pis_plate_variantv	Array of LP recognition result variants.
pis_outer_rect	Rectangular area (containing the LP image) which was specified relative to the image. The field type (vodi_rect_t) represents a rectangle with orthogonal sides. This field is set when the basic LP detection is performed on the frame, and, thus, its value is not highly precise (for example, this rectangular area may cover vehicle lights, radiator etc.).
pis_exact_rect	Rectangular area (containing the LP image) which was specified relative to the image. The field type (vodi_rect_t) represents a rectangle with orthogonal sides. This parameter's value is more precisely calculated than **pis_outer_rect**'s one, thus, it covers only the LP symbols. This field is used for backward compatibility and the **pis_exact_quad** parameter is now used instead.
pis_exact_quad	Rectangular area (containing the LP image) which was specified relative to the image. This field has higher value precision than **pis_outer_rect** and **pis_exact_rect**. This field type (vodi_quad64f_t) represents the LP angle on the image.
pis_inversed	Flag of inversed number.
pis_direction	Direction of the vehicle movement.
pis_angle	LP angle on the frame.
pis_speed	LP movement speed. Reserved.
pis_index	LP identifier.
pis_flags	Available flags:
	VodiF_RESULT_DUP
	VodiF_RESULT_LOST
	VodiF_RESULT_INVALID
pis_move_in	Reserved.
pis_status	Reserved.
pis_uctx	User's context data.
pis_image	LP image.
pis_bftime	Timestamp of the best frame containing the current LP.
pis_ctime	Timestamp of the frame on which the current LP was recognized for the first time (which determines the vehicle appearance on the frame).
pis_ltime	Timestamp of the frame on which the current LP was recognized last time (which determines that the vehicle left the surveillance zone).

Parameter	Description
pis_imgsz	Reserved.
pis_seqnum	Reserved.

6.6 vodi_plate

6.6.1 Name

struct vodi_plate — license plate.

6.6.2 Syntax

```
#include <Bo/aorp/Vpwtypes.h>

struct vodi_plate {
    wchar_t             pv_plate_string[VodiL_VPW_SYMBOL_MAX + 1];
    wchar_t             pv_plate_type[VodiL_VPW_SYMBOL_MAX + 5];
    u_int32_t           pv_plate_id;
    u_int32_t           pv_country_id;
    u_int32_t           pv_subdivision_id;
    float               pv_validity;
    float               pv_validity_coeff;

    float               pv_min_symb_validity;
    float               pv_max_geom_deviation;
    float               pv_max_symb_validity;
    float               pv_avg_symb_validity;

    vodi_plateid_t      pv_tmpl_id;

    vodi_color_t        pv_background;
    vodi_color_t        pv_sym_color;
    size_t              pv_symbolc;
    vodi_plate_symbol_t pv_symbolv[VodiL_VPW_SYMBOL_MAX];
    void_rect_t         pv_exact_rect;
};
```

6.6.3 Description

Table 17: struct vodi_plate parameters

Parameter	Description
pv_plate_string	Unicode string containing the license plate number.
pv_plate_type	License plate type, specified by string:
	z lower-case letter
	Z upper-case letter
	x number (0-9)
	| space between symbols

Parameter	Description
	A digit (letter or number)
pv_plate_id	Identifier of the appropriate LP template.
pv_country_id	LP issuer-state code (in accordance with ISO 3166-1 numeric).
pv_subdivision_id	Reserved.
pv_tmpl_id	Identifier of the LP template (type — vodi_plateid_t), which includes the template number, country code and subdivision code (if any) according to ISO 3166. This field is provided to be used instead of pv_plate_id, pv_country_id, pv_subdivision_id fields, which are considered obsolete. See the Description chapter for detailed description of the functions which operate the vodi_plateid_t structures.
pv_validity	Precision of the LPR result correctness (in accordance with the **AutoSDK** internal criteria).
pv_validity_coeff	Reserved.
pv_min_symb_validity	Reserved.
pv_avg_symb_validity	Reserved.
pv_max_geom_deviation	Reserved.
pv_avg_geom_deviation	Reserved.
pv_background	LP background color.
pv_sym_color	LP symbols color.
pv_symbolc	Number of symbols (on the LP).
pv_symbolv	Array of symbols (from the LP).
pv_exact_rect	Rectangular area (containing the LP image) which was specified relative to the image. The area covers only the LP symbols.

6.7 aorp_error

6.7.1 Name

struct aorp_error — error description.

6.7.2 Syntax

```
#include <Bo/aorp/Types.h>

struct aorp_error {
    char const   *func;
    char const   *file;
    long          line;
    bo_status_t   status;
    syserrcode_t  syserr;
    unsigned      msgidx;
    unsigned      bufsz;
    char         *msg;
};
```

6.7.3 Description

Table 18: struct aorp_error parameters

Parameter	Description
func	Name of the function that caused an error. The field is optional and is usually set in debug versions of the program.
file	Path to the file containing the function which triggered the error. The field is optional and is usually set in debug versions of the program.
line	Line number in the **file** where error had appeared. The field is optional and makes sense only if the **file** parameter is not NULL.
status	Error code and domain (see the <Bo/Errors.h> file).
syserr	Current OS error code. If OS error codes usage is not necessary, POSIX error codes can be used instead.
msgidx	Sequential number of the message (see the <Bo/SMessages.h> file).
bufsz	Size of the buffer (in bytes) which is pointed by **msg**.
msg	Pointer to string that describes an error.

7

DESCRIPTION OF OPERATIONS WITH TEMPLATES

- The folders in **doc/templates** directory contain images of LPs of various types.
- The LPs of each set were issued in the country stated in folder name.
- The LP images names are equivalent to order numbers of corresponding LP templates within the country module (**vpwi-[country-ID]**[1]).

The **doc/templates/templates_map.txt**[2] file describes the conformance between the former LP template identifiers used by **AutoSDK** and the new ones[3].
Record format: CountryId[.SubdivisionID].TemplateId:OldTemplateID
Also, to print information on all or specific LP templates, use the **lsvpwi** CLI application.

7.1 vodiiso3166children

7.1.1 Name

VodiISO3166children — gets the subdivisions list for the specified country.

7.1.2 Syntax

```
#include <Vodi/ISO3166lib.h>

struct vodi_iso3166_elem const **
__attribute__((nonnull(1)))
VodiISO3166children(
    struct vodi_iso3166_elem const *anElem
    );
```

7.1.3 Description

The **VodiISO3166children** function's argument is the pointer to vodi_iso3166_elem structure, which represents a specific country (in accordance with ISO 3166-1 standard).

[1] for example, the "vpwi-co" module, which is used for recognition of Colombian LPs
[2] available since 2.5.8 version
[3] available since 2.5.1 version

7.1.4 Return values

In case of success, the **VodiISO3166children** function returns a NULL-terminated array of pointers to vodi_iso3166_elem structures. Those structures will represent subdivisions of a country specified by the argument (in accordance with ISO 3166-2).

7.2 lpvlibshowplateid

7.2.1 Name

LpvlibShowPlateid — converts the structure identifying the LP template into a C-string.

7.2.2 Syntax

```
#include <Vodi/services/Lpvlib.h>

char *
__attribute__((nonnull(1)))
LpvlibShowPlateid(
    char aResultPtr[/* LpvlibK_PLATEID_CHAR_MAX */],
    vodi_plateid_t aPlateid,
    aorp_opflags_t Flags
    );
```

7.2.3 Description

The **LpvlibShowPlateid** function has the following parameters:

- **aResultPtr** — buffer which was prepared for the resulting string.
- **aPlateid** — the vodi_plateid_t structure which C-string representation is needed.
- **Flags** — needed resulting string format: if **LpvlibF_HUMAN_READABLE** flag is passed, you will get the alphabetic representation of the country/subdivision codes; if 0 is passed — the numeric one.

7.2.4 Return values

In case of success, the **LpvlibShowPlateid** function will return the pointer to a string that describes the template specified by the argument. Otherwise, the function will return NULL.

7.3 lpvlibreadplateid

7.3.1 Name

LpvlibReadPlateid — create a structure identifying the LP template from a C-string.

7.3.2 Syntax

```
#include <Vodi/services/Lpvlib.h>

vodi_plateid_t
__attribute__((nonnull(1)))
LpvlibReadPlateid(
    char const *aSrc,
    struct aorp_error *anErrPtr
    );
```

7.3.3 Description

The **LpvlibReadPlateid** function receives the following parameters:

- **aSrc** — pointer to a C-string which contains the description of specific LP template.
- **anErrPtr** — pointer to aorp_error structure (the **anErrPtr** parameter).

7.3.4 Return values

In case of success, the **LpvlibReadPlateid** function will return the vodi_plateid_t structure. The structure will contain the data read from the C-string passed as an argument (if the string itself is syntactically correct). Otherwise the function will return the **VodiK_INVAL_PLATEID** value and will write the error description by the **anErrPtr** parameter.

7.4 lpvlibmodulesof

7.4.1 Name

LpvlibModulesOf — get the list of vpwi modules which contain templates specified by aSet parameter; identify which minimal set of modules to load in order to use a specific templates set.

7.4.2 Syntax

```
#include <Vodi/services/Lpvlib.h>

bo_seqlist_(char *) *
__attribute__((nonnull(2)))
LpvlibModulesOf(
    aorp_object_t aSelf /* = NULL */,
    bo_seqlist_(char *) *aResultPtr,
    char const *aSet /* = NULL */,
    struct aorp_error *anErrPtr
    );
```

7.4.3 Description

The **LpvlibModulesOf** function receives the following parameters:

- **aSet** — string representation of templates set.
- **aResultPtr** — pointer to resulting list.
- **anErrPtr** — pointer to the structure which describes an error.

Example of a function which loads the needed vpwi modules in accordance with the templates set:

```
#include <Bo/fundis/Seqlist.h>
#include <Vodi/services/Lpvlib.h>

static bo_status_t
load_modules(
    char const *aTemplates,
    _Bool Force,
    struct aorp_error *anErrPtr
    )
{
    bo_status_t status;
    bo_seqlist_t mdlst;

    /* loading the vpwi module which contains the implementation of LpvlibModulesOf */
    status = AorpMldLoad("vpwi", NULL, 0, false, anErrPtr);
```

```
16      if (BoS_FAILURE(status)) {
17          return (status);
18          /* NOTREACHED */
19      }
20
21      status = BoS_ERR;
22      BO_SEQLIST_Init(&mdlst);
23      if (NULL != LpvlibModulesOf(NULL, &mdlst, aTemplates, anErrPtr)) {
24          BO_SEQLIST_FOREACH_(char *, &mdlst)
25              status = AorpMldLoad(*_$1, NULL, 0, false, anErrPtr);
26              if (BoS_FAILURE(status) && !Force)
27                  break;
28          FOREACH_END
29      }
30      LpvlibStrlstDestroy(&mdlst);
31
32      return (Force ? BoS_NORMAL : status);
33  }
```

7.4.4 Return values

In case of success, the **LpvlibModulesOf** function will return a aResultPtr pointer. Otherwise the function will return NULL and will write the error description by the **anErrPtr** parameter.

8

DESCRIPTION OF GRAMMAR FOR SPECIFYING THE TEMPLATES SETS

The grammar is used by the following functionality:

- functions of the Lpvlib family;
- VodiprincControl function's commands:
  - VodiCTL_GETVPWI
  - VodiCTL_ADDVPWI
  - VodiCTL_SETVPWI
  - VodiCTL_DELVPWI
- vpwfetch utility (as an argument of -e, --templates option);
- lsvpwi utility;
- lsvpwc utility.

```
1   E    : T1 \ ... \ Tn          (1 <= n, left associative) ;
2   T    : F1 ... Fn              (0 <= n) ;
3   F    : * | pse | ( E ) | ! F ;
4
5   pse  : oid | oid . cs | cs ;
6   oid  : code1 . ... . coden    (1 <= n) ;
7   cs   : [crs] | [^crs] ;
8   crs  : cr1 , ... , crn        (0 <= n) ;
9   cr   : code | code - code | code - inf ;
10  code : INT | ISO3166-name ;
```

- E (*expression*) — start variable/symbol.
- \ — operator of templates set difference.
- **T** — subsets which are implicitly united with a space.
- * — universal set.
- *pse* — set of plate IDs.
- *oid* — set of numerical country codes.
- *cs* — codes set.

- *crs* — set of code ranges.
- *cr* — code range.

You can use either literal or numerical ISO3166 country codes to specify template sets (for example, "ua.3", "804.3").
Examples:

- **784.3** — select all the supported LP templates which belong to Dubai, UAE.
- **784.3.**[**1-2**] or **ae.du.**[**1,2**] — select the first two LP templates which belong to Dubai, UAE.
- **ae.du.**[**3,1**] or **784.3.1 784.3.3** — select LP templates of Dubai, UAE, with the sequence numbers 1 and 3.
- **ua** — select all the LP templates of Ukraine.
- **784 804** or **ua ae** — select all the LP templates of Ukraine and UAE.
- [**1-19**] [**31-inf**] — select LP templates of all the suported countries, which codes are in range 20-30.
- **784.**[**1-2**] **784.3.**[**11-inf**] **784.**[**4-inf**] or **ae\ae.du.**[**1-10**] — select all the templates of UAE, except for Dubai templates with sequence numbers in range 1-10.
- **!zm** or [**1-893**] [**895-inf**] — select all the available LP templates, except for those that belong to Zambia.
- **!(zm zw)** or [**1-715**] [**717-893**] [**895-inf**] — select all the available LP templates, except for those that belong to Zambia and Zimbabwe.