JAI-PULNiX Intelligent Transportation Systems

Vehicle Imaging System 300/400
(VIS 300 - VIS 400)

ImageServer User’s Guide
Notice

The material contained in this manual consists of information that is proprietary to JAI PULNiX, Inc., and may only be used by the purchasers of the product. JAI PULNiX, Inc. makes no warranty for the use of its product and assumes no responsibility for any errors which may appear or for damages resulting from the use of the information contained herein. JAI PULNiX, Inc. reserves the right to make changes without notice.

Microsoft, Windows XP, Windows 2000, Windows 98, Windows 95, Windows NT, and Windows Explorer are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Warranty

All of our solid-state cameras sold in North America have a full three-year warranty. Those sold elsewhere have a full one-year warranty. If any such product proves defective during the warranty period, JAI PULNiX, Inc. will repair the defective product without charge for parts and labor or will provide a replacement in exchange for the defective product. This warranty shall not apply to any damage, defect or failure caused by improper use or inadequate maintenance.

ImageServer User’s Guide

JAI PULNiX, Inc.
625 River Oaks Parkway
San Jose, CA 95134
www.jai.com

<table>
<thead>
<tr>
<th>United States of America</th>
<th>United Kingdom</th>
<th>Other Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Department, direct phone: +1 408 383-0800</td>
<td>Sales Department, direct phone: +44 (0) 189 582-1481</td>
<td>Sales Department, direct phone: +45 4457 8888</td>
</tr>
<tr>
<td>Toll free: 800 445-5444</td>
<td>Technical Support, direct phone: +44 (0) 771 412 6293</td>
<td></td>
</tr>
<tr>
<td>Technical Support, direct phone: +1 408 383-4393</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Table of Contents

1 Document Overview ............................................. 1  
  1.1 List of Abbreviations ........................................... 1

2 ITS ImageServer Functionality ..................................... 2  
  2.1 ImageServer Functions .......................................... 3
  2.2 IS Configuration Examples ...................................... 4  
    2.2.1 Section Control ............................................ 4
    2.2.2 ALPR to DB ................................................. 5

3 Installation ......................................................... 6

4 Configuration ....................................................... 7

5 Operational Description ........................................... 9  
  5.1 ImageServer Window ............................................. 9  
    5.1.1 File Menu ................................................... 9
    5.1.2 View Menu .................................................. 10
    5.1.3 Test Menu .................................................. 10
    5.1.4 Image Preview Window .................................... 11
    5.1.5 Image Properties Window ................................ 12

6 Output ............................................................. 14  
  6.1 Image tags ..................................................... 14
  6.2 VDT files ....................................................... 14

7 Interfaces .......................................................... 17  
  7.1 Process Server interface ...................................... 17
  7.2 Lane Controller interface ..................................... 24
List of Figures

FIGURE 1. ImageServer flow chart .................................................. 2
FIGURE 2. ImageServer Section Control flow chart .......................... 4
FIGURE 3. ImageServer adding ALPR to the database ...................... 5
FIGURE 4. IS Window ........................................................................ 9
FIGURE 5. File Menu selections ...................................................... 9
FIGURE 6. Picture Window .............................................................. 10
FIGURE 7. Test and Load images .................................................... 10
FIGURE 8. The Image Preview ....................................................... 11
**List of Tables**

| TABLE 1.   | Configuration                                    | 7 |
| TABLE 2.   | Image Properties                                 | 12 |
| TABLE 3.   | Image Properties continued                       | 13 |
| TABLE 4.   | ImageServer ITS tags table                       | 14 |
| TABLE 5.   | VDT content                                      | 16 |
JAI-PULNiX Intelligent Transportation Systems

Vehicle Imaging System 300/400
(VIS 300 - VIS 400)

ImageServer Users Guide

1 Document Overview

This document describes the Image Server program. The functionality described in this document is based on released version 1.0.0 of ImageServer.

The Image Server program is used to collect and process images from several EN cameras.

The ImageServer (IS) is designed as a gateway between the JAI EN camera and whatever application the end customer may have.

Terminology:

“EN Camera” is the JAI PULNiX TS(C)-9720EN and TS(C)-2030EN camera.

“PC” is the device that runs the application.

1.1 List of Abbreviations

- IS    ImageServer
- VDT   Vehicle Detection Tag
- ALPR  Automatic License (number) Plate Recognition
- DB    Data Base
- ADO   ActiveX Data Objects
- LC    Lane Controller
- HASP  Hardware Against Software Piracy
2 ITS ImageServer Functionality

The IS can receive Images directly from EN cameras by means of their Ethernet connections; or from an intermediate file store (folder), where the attached cameras can automatically place the images through FTP.

The IS can also be configured to do various image pre-processing before passing them on to file storage, database, or protocol. Pre-processing could be plate finding, plate reading, fingerprint generation and so on. Figure 1 shows the full potential of the IS.

**FIGURE 1. ImageServer flow chart**
2.1 ImageServer Functions

The ImageServer can:

• Collect images from the TS(C)-9720EN or TS(C)-2030EN camera or watch a folder for incoming image files. One IS application can do either or both at the same time for up to 16 cameras/folders, depending on the PC performance.

• Receive extra information about the vehicle from a lane controller. The information can be correlated with the image based on the time-stamp in the image and in the message from the lane controller. The lane controller information can be tagged into the image and also the VDT.

• Processed images are tagged with relevant information like ImageServer ID, Site ID, and lane number. Images and tags can be saved as TIFF or JPEG.

• Combine several images into one VDT. For instance, images from the front and the back of a vehicle can be stored together as one VDT.

• Plate finding. IS can run a plate finder, and based on the result cut out a plate image and store that together with the original image.

• Fingerprints can be generated and stored in the VDT.

• Automatic License Plate Reading (ALPR). Read the license plate in the Image and tag the result into the image and store it in the VDT.

• Send the results to a ProcessServer over TCP/IP

• Store results directly into a database

• Store the resulting files in a folder

The IS output consists of a VDT in the form of an XML file, and one or more images tagged with information, optionally including an image of the license plate.

The ImageServer runs on a Windows XP or Windows Server 2003 PC. It can run as a Windows service or as a normal application.

The ImageServer functionality can be configured, making it a very versatile tool.
2.2 IS Configuration Examples

2.2.1 Section Control

FIGURE 2. ImageServer Section Control flow chart
2.2.2 ALPR to DB

An example of an alternative use of the ImageServer would be if only wanted to add license plate recognition (ALPR) to the images and store the results in a database. The tasks are directed as shown in Figure 3.

**FIGURE 3. ImageServer adding ALPR to the database**
3 Installation

Before installing the ImageServer verify the Microsoft .NET Framework software version 1.1 and 2.0 are installed on the system. Download the software is at http://www.Windowsupdate.com.

After the .NET framework is installed, install the ImageServer by copying the ImageServer folder to the local hard disk. Normally the local address would be “C:\Program Files\JAI A-S\ImageServer”.

The ImageServer is protected by a HASP dongle.

- If the computer is connected to the internet, you can insert the dongle into an USB port of the computer, and the driver for the HASP dongle installs automatically.
- If the computer is not connected to the internet, the HASP dongle driver can be downloaded from http://www.aladdin.com/ and then installed on the computer.
4 Configuration

The ImageServer is configured by the settings file ImageServerSettings.xml, located in the same folder as the ImageServer program.

**TABLE 1. Configuration**

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerName&gt;</td>
<td>The name of the ProcessServer where you connect. Leave empty if the ImageServer should not connect to a ProcessServer.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerPort&gt;</td>
<td>The port to use when connecting to the ProcessServer.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerMessageTimeout&gt;</td>
<td>The timeout in seconds to wait for a message from the ProcessServer.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerMessageRetry&gt;</td>
<td>The number of retries when sending messages to the ProcessServer.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerTransmitMode&gt;</td>
<td>The startup transmit mode of the ProcessServer interface.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerIFLogFile&gt;</td>
<td>Log the ProcessServer communication to a file.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerIFCheckConstraints&gt;</td>
<td>Check the ASN.1 constraints of the messages on the ProcessServer interface.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerIFMaxLogFileSizeKB&gt;</td>
<td>The size of the ProcessServer interface log files.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ProcessServerIFNumberOfLogFile&gt;</td>
<td>The number of logfiles to keep for the ProcessServer interface.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;LaneListenPort&gt;</td>
<td>Lane Controller listen port.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;lane_controller_msg_timeout&gt;</td>
<td>Lane Controller message timeout in milliseconds.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;lane_controller_receive_timeout_ms&gt;</td>
<td>Lane Controller receive timeout in milliseconds.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ImageServerID&gt;</td>
<td>ImageServer ID, used as the ImageServer ID when tagging the images.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;MaximumFiles&gt;</td>
<td>The maximum number of files to keep on the disk.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ImageSaveType&gt;</td>
<td>The image format to use when saving images, TIFF or JPEG is supported.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;ImageRootPath&gt;</td>
<td>The root folder where the ImageServer stores the images.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;LogLevel&gt;</td>
<td>The LogLevel (0 to 100).</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;LogFile&gt;</td>
<td>Save ImageServer log to a file.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;MaxLogFileSize&gt;</td>
<td>The max size of the logfiles in kb.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;NumberOfLogfiles&gt;</td>
<td>The number of logfiles to keep.</td>
</tr>
<tr>
<td>&lt;Settings&gt;</td>
<td>&lt;Lanes&gt;</td>
<td>The lanes the IS is collecting images from. This is a grouping, so lanes can hold several lane settings.</td>
</tr>
<tr>
<td>&lt;Settings&gt;Lanes</td>
<td>&lt;Lane&gt;</td>
<td>One lane with the settings of this lane.</td>
</tr>
<tr>
<td>&lt;Settings&gt;Lanes</td>
<td>&lt;site_id&gt;</td>
<td>The site id of the lane.</td>
</tr>
<tr>
<td>&lt;Settings&gt;Lanes</td>
<td>&lt;lane_number&gt;</td>
<td>The lane number.</td>
</tr>
<tr>
<td>&lt;Settings&gt;Lanes</td>
<td>&lt;data_collect&gt;</td>
<td>Collect data from this lane.</td>
</tr>
<tr>
<td>&lt;Settings&gt;Lanes</td>
<td>&lt;get_lc_messages&gt;</td>
<td>Get and wait for lane controller messages for this lane, set only to true if you have a lane controller connected to the ImageServer.</td>
</tr>
<tr>
<td>&lt;Settings&gt;Lanes</td>
<td>&lt;cameras&gt;</td>
<td>The cameras for this lane. This is a grouping, so cameras can hold several camera settings.</td>
</tr>
<tr>
<td>&lt;Settings&gt;Lanes</td>
<td>&lt;Camera&gt;</td>
<td>One camera with the settings for this camera.</td>
</tr>
</tbody>
</table>
### TABLE 1. Configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;camera_id&gt;</code></td>
<td>The camera ID</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;camera_name&gt;</code></td>
<td>The camera name, or the IP address e.g 10.0.0.100</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;make_matchdata&gt;</code></td>
<td>Generate Match data from the images taken with this camera</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;extract_license_plate&gt;</code></td>
<td>Extract a license plate image from the images taken with this camera</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;read_license_plate&gt;</code></td>
<td>Try to read the license plate from the images taken with this camera</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;folder_path&gt;</code></td>
<td>If this camera is a folder on the hard disk, look for images in this folder</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;folder_filter&gt;</code></td>
<td>The filter to use when looking for images in a folder, e.g. '*.tif'</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;get_images_from_folder&gt;</code></td>
<td>If set to true collect images from a folder, else collect direct from a camera</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;get_errors_direct_from_camera&gt;</code></td>
<td>Relay the error messages from the camera to the ProcessServer</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;use_license_plate_reader_as_finder&gt;</code></td>
<td>Use the license plate reader also as the license plate finder</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Settings&gt;&lt;Lanes&gt;</code></td>
<td><code>&lt;generate_JAI_Matcher_matchdata&gt;</code></td>
<td>Generate JAI_Matcher SDK data and not the old format.</td>
</tr>
<tr>
<td><code>&lt;Lane&gt;&lt;cameras&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Camera&gt;</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5 Operational Description

5.1 ImageServer Window

The IS window displays software activity details. Information includes the image index, time, log level and text.

FIGURE 4. IS Window

5.1.1 File Menu

- The command File=>Save Settings saves the settings file ImageServerSettings.xml with the current used settings.
- The command File=>Exit closes the program.

FIGURE 5. File Menu selections
5.1.2 View Menu

The command “View=>Picture Window” activates the picture window.

The picture window displays the latest image received and processed by the ImageServer. The Picture window is only updated when the window is displayed, so it may be empty when opened the first time.

**FIGURE 6. Picture Window**

5.1.3 Test Menu

The command “Test=>Load Images” allows the user to test the ImageServer by selecting images from files. The Images are loaded into the ImageServer as if they came from the first lane defined in the ImageServerSettings.xml file. Use this menu only for test purposes.

**FIGURE 7. Test and Load images**
5.1.4 Image Preview Window

The “Image Preview” window displays the latest image processed by the ImageServer. Additional information about the image is also displayed at the top, and the plate reader information at the bottom. The license plate is displayed at the bottom left if any plate was found in the image. At the bottom right the plate reader information displays with the plate read, the plate country and a confidence percentage.

Full information in the image can be seen in the Image Properties Window, which can be opened from the View menu in the Image Preview window.

The display of the Image Preview windows requires additional CPU power so it should only be open for demonstration or debug purpose. The ImageServer always starts up with the Image Preview windows closed.

FIGURE 8. The Image Preview
5.1.5 Image Properties Window

The information in the Image Properties window are mostly from the EN camera, the ITS ImageServer Tags section contains the information the ImageServer have added to the Image. This is similar to the information that the VDT contains.

<table>
<thead>
<tr>
<th>TABLE 2. Image Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property</strong></td>
</tr>
<tr>
<td>Date and Time</td>
</tr>
<tr>
<td>Date/TimeOriginal</td>
</tr>
<tr>
<td>SubsecTime</td>
</tr>
<tr>
<td>SubsecTimeOriginal</td>
</tr>
<tr>
<td>EXIF Version Type</td>
</tr>
<tr>
<td>Image Data Structure</td>
</tr>
<tr>
<td>Image Width</td>
</tr>
<tr>
<td>Image Length</td>
</tr>
<tr>
<td>BitsPerPixel</td>
</tr>
<tr>
<td>Compression</td>
</tr>
<tr>
<td>PhotometricInterpretation</td>
</tr>
<tr>
<td>StripOffsets</td>
</tr>
<tr>
<td>SamplesPerPixel</td>
</tr>
<tr>
<td>RowsPerStrip</td>
</tr>
<tr>
<td>StartOfScan</td>
</tr>
<tr>
<td>XResolution</td>
</tr>
<tr>
<td>YResolution</td>
</tr>
<tr>
<td>ResolutionUnit</td>
</tr>
<tr>
<td>ColorMap</td>
</tr>
<tr>
<td>Information</td>
</tr>
<tr>
<td>Manufacturer</td>
</tr>
<tr>
<td>Model</td>
</tr>
<tr>
<td>Software</td>
</tr>
<tr>
<td>Copyright</td>
</tr>
<tr>
<td>ImageUniqueID</td>
</tr>
<tr>
<td>ITS Camera Tags</td>
</tr>
<tr>
<td>ImageCRC32</td>
</tr>
<tr>
<td>LSSideA</td>
</tr>
<tr>
<td>LSSideB</td>
</tr>
<tr>
<td>CDSGain</td>
</tr>
<tr>
<td>AsymShutterMode</td>
</tr>
<tr>
<td>xHeight</td>
</tr>
<tr>
<td>yTop</td>
</tr>
<tr>
<td>NightLight</td>
</tr>
<tr>
<td>ExposureNumber</td>
</tr>
<tr>
<td>ImageType</td>
</tr>
<tr>
<td>EstimatedNTPError</td>
</tr>
<tr>
<td>NTStatus</td>
</tr>
<tr>
<td>ModelType</td>
</tr>
<tr>
<td>LaserVehicleLength</td>
</tr>
<tr>
<td>LaserMinimumRange</td>
</tr>
<tr>
<td>LaserVehicleHeight</td>
</tr>
</tbody>
</table>
### TABLE 3. Image Properties continued

<table>
<thead>
<tr>
<th>ImageServer Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDTNNumber</td>
<td>2041400</td>
</tr>
<tr>
<td>ServerID</td>
<td>1</td>
</tr>
<tr>
<td>SiteID</td>
<td>0</td>
</tr>
<tr>
<td>LaneNumber</td>
<td>0</td>
</tr>
<tr>
<td>FileID</td>
<td>2041400</td>
</tr>
<tr>
<td>CameraID</td>
<td>1</td>
</tr>
<tr>
<td>PlateFinderStatus</td>
<td>Plate Found</td>
</tr>
<tr>
<td>PlateTypeFound</td>
<td>No Plate Type</td>
</tr>
<tr>
<td>PlateNominalCenterX</td>
<td>315</td>
</tr>
<tr>
<td>PlateNominalCenterY</td>
<td>141</td>
</tr>
<tr>
<td>PlateRegionX</td>
<td>306</td>
</tr>
<tr>
<td>PlateRegionY</td>
<td>113</td>
</tr>
<tr>
<td>PlateRegionHeight</td>
<td>18</td>
</tr>
<tr>
<td>LicensePlate</td>
<td>VZ25734</td>
</tr>
<tr>
<td>LicensePlateCountry</td>
<td>OK</td>
</tr>
<tr>
<td>LicensePlateConfidence</td>
<td>722</td>
</tr>
</tbody>
</table>

**Plate Nominal Center X**

Plate Nominal Center X
6 Output

6.1 Image tags

When the ImageServer stores images it tags the images with information that the ImageServer is possessing.

A description of the image tags the EN camera tags the image with can be found in the LC to EN interface document. Here it is also described how the Custom ITS tags are stored.

The ImageServer tags are custom ITS tags.

### TABLE 4. ImageServer ITS tags table

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFE100</td>
<td>VDTNumber</td>
<td>VDT Number</td>
</tr>
<tr>
<td>0xFE101</td>
<td>ServerID</td>
<td>Server ID</td>
</tr>
<tr>
<td>0xFE102</td>
<td>SiteID</td>
<td>Site ID</td>
</tr>
<tr>
<td>0xFE103</td>
<td>LaneNumber</td>
<td>Lane Number</td>
</tr>
<tr>
<td>0xFE104</td>
<td>FileID</td>
<td>File ID</td>
</tr>
<tr>
<td>0xFE105</td>
<td>CameraID</td>
<td>Camera ID</td>
</tr>
<tr>
<td>0xFE106</td>
<td>PlateFinderStatus</td>
<td>Plate Finder Status</td>
</tr>
<tr>
<td>0xFE107</td>
<td>PlateTypeFound</td>
<td>Plate Type Found</td>
</tr>
<tr>
<td>0xFE108</td>
<td>PlateNominalCenterX</td>
<td>Plate Nominal Center X</td>
</tr>
<tr>
<td>0xFE109</td>
<td>PlateNominalCenterY</td>
<td>Plate Nominal Center Y</td>
</tr>
<tr>
<td>0xFE10A</td>
<td>PlateRegionX</td>
<td>Plate Region X</td>
</tr>
<tr>
<td>0xFE10B</td>
<td>PlateRegionY</td>
<td>Plate Region Y</td>
</tr>
<tr>
<td>0xFE10C</td>
<td>PlateRegionWidth</td>
<td>Plate Region Width</td>
</tr>
<tr>
<td>0xFE10D</td>
<td>PlateRegionHeight</td>
<td>Plate Region Height</td>
</tr>
<tr>
<td>0xFE10E</td>
<td>LaneControllerData</td>
<td>Lane Controller Data</td>
</tr>
<tr>
<td>0xFE10F</td>
<td>LicensePlate</td>
<td>License Plate</td>
</tr>
<tr>
<td>0xFE110</td>
<td>LicensePlateCountry</td>
<td>License Plate Country</td>
</tr>
<tr>
<td>0xFE111</td>
<td>LicensePlateConfidence</td>
<td>License Plate Confidence</td>
</tr>
<tr>
<td>0xFE112</td>
<td>LicensePlateCharConfidence</td>
<td>License Plate Char Confidence</td>
</tr>
</tbody>
</table>

6.2 VDT files

The VDT is saved in a XML file, a sample of a VDT file is shown below:

```xml
<?xml version="1.0" encoding="utf-8" ?>
    <vdt_number>2042772</vdt_number>
    <server_id>1</server_id>
    <site_id>0</site_id>
    <lane_number>0</lane_number>
    <image_status>
        <ImageStatus>
            <file_id>2042772</file_id>
            <camera_id>1</camera_id>
        </ImageStatus>
    </image_status>
    <vdt_status>
        <missing_lanecontroller_data>true</missing_lanecontroller_data>
    </vdt_status>
</VDT>
```
<plate_finder_status>PlateFinderStatus_plate_found</plate_finder_status>
<plate_type_found>PlateType_no_plate_type</plate_type_found>
<nominal_centre_x>270</nominal_centre_x>
<nominal_centre_y>276</nominal_centre_y>
<plate_region_x>211</plate_region_x>
<plate_region_y>267</plate_region_y>
<plate_region_width>119</plate_region_width>
<plate_region_height>19</plate_region_height>
<license_plate>TP91078</license_plate>
<license_plate_country>DK</license_plate_country>
<license_plate_confidence>937</license_plate_confidence>
</vdt_status>

<match_data>DgEUAZyKAAAAAAADQhEd0iAQBAAEAAAAAAAAAMIAAgAACAACAAAAAAAAADAAhAA0FAEOAq5BCwFF 
AQAAAAAAAAAAABwEGAAYABwAHAAAcAB/ABH8AcABwAHAAAcABeAM8AxwDHAMcA3wB+AHgAAAAAAAAA 
AAOxw/j/+P44/DhMOA44Djj+OP44/DBsMca5ZDmOGY4ZZDnAMMAw4PF/4H/AHAACax8OBw4HA 
gMCA4MHg4cD/whVhAYADgAcA8ADwAPADcABwAHAAAcABwAHAAAcABwAHAAAcABwAHAAAcADg8GH4f/f/9+H4z4fDB8A 
HwA///D/8e/ABDgEOAw4DDg00Aw4DDgMOA44Djg00Awxbp/A4OHA4cDhwOHA8cBxwOHA4cDhwODj4 ///j/ 
cABwAGAA4ADgAcABwAGA4ADgAcOwAeDn4f/hj8MPww/DD8MPww/Dh8OHg4cHAAcAbwP/w ///H54fj800xw7HHe4czj/wBAECAQf9D/OP/84AAAAAAA/4f/j///://AAAAADAAfD//////////HQA AAB8AP/ 
H/////////8AAAAAAhAfp/////////8AAAAAwA8AP/////////8AAAAAAhAfp/////////8AAAAAAhAfp/// // 
z8HAAAYAbgAAAg/d///BwADAAAMMMMMMf///w8PDGcAAAAAADM3///8BfwAAAAAAA/A+IGfAA4I 
AAAAADwffffff///H+AAAGAAAAAdf///BwABAAAADAAH///7/f+AOQA5ADgAOAAA4AdgAOAA4AdyAPgA+A 
DyDwHPef8R//whHAAscBxwHAscBxwHAcB2zp/B/CaIA8DDDCWHgFOAf4Dh4cDjwOPAPgA+ABw+H 
D4cPHw8PwB/AD4CH8cPw8PIAAIAAgAgc/wbWCAAAAAAAAa/8AAAAAAAIR/z8APwB8 
ABAABAA/v8ACABwAADAAAAAAA0PPAf4B8AAAAAAAAABwAPAA4AAAAAA48Ph48D/ADwAAAAABw 
Hnke/x///HwAAACG44/////////wAAAAAAAGAAAA90eAAADzdOB88f///iAAABBgEGAYf///AAAAHwQfADuD/// 
///39jOMw42Dn8///+HBwcAD4Ad4f4P///8</match_data>

<program_version>
<int>1</int>
<int>0</int>
<int>20295</int>
</program_version>

<date_time>2006-03-10T11:17:47.453000+01:00</date_time>
<img_srv_check_sum>ZF+Gvsc+cfUwCovPnmClIQ==</img_srv_check_sum>
### TABLE 5. VDT content

<table>
<thead>
<tr>
<th>Section</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;vdt_number&gt;</td>
<td>The VDT identifier</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;server_id&gt;</td>
<td>The ImageServer identifier</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;site_id&gt;</td>
<td>Site identifier of lane</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;lane_number&gt;</td>
<td>Lane identifier</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;match_data&gt;</td>
<td>Match data (fingerprint)</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;program_version&gt;</td>
<td>ProgramVersion, -- program version of ImageServer</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;date_time&gt;</td>
<td>Date time of VDT from camera</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;img_srv_check_sum&gt;</td>
<td>Checksum of lanecontroller executable (md5hash)</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;image_status&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;file_id&gt;</td>
<td>The file identifier</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;camera_id&gt;</td>
<td>Camera identifier</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;missing_lanecontroller_data&gt;</td>
<td>The lane controller data is missing</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;plate_finder_status&gt;</td>
<td>The status from the plate finder</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;nominal_centre_x&gt;</td>
<td>The center of the plate x coordinate</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;nominal_centre_y&gt;</td>
<td>The center of the plate y coordinate</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;plate_region_x&gt;</td>
<td>The plate region x coordinate</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;plate_region_y&gt;</td>
<td>The plate region y coordinate</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;plate_region_width&gt;</td>
<td>The plate region width</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;plate_region_height&gt;</td>
<td>The plate region height</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;license_plate&gt;</td>
<td>The license plate read</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;license_plate_country&gt;</td>
<td>The country of the license plate</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;license_plate_confidence&gt;</td>
<td>The confidence of the license plate (0-1000)</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;date_time&gt;</td>
<td>Date and time from the lanecontroller</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;site_id&gt;</td>
<td>Site identifier</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;lane_number&gt;</td>
<td>Lane identifier</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;data&gt;</td>
<td>The lanecontroller data</td>
</tr>
<tr>
<td>&lt;VDT&gt;</td>
<td>&lt;image_server_date_time&gt;</td>
<td>Date and time when lane controller message arrived in ImageServer, local time of the ImageServer</td>
</tr>
</tbody>
</table>
7 Interfaces

7.1 Process Server interface

The interface to the process server is defined using ASN.1 notation (Abstract Syntax Notation One). The ImageServer uses an open source ASN.1 compiler available at [http://lionet.info/asn1c/](http://lionet.info/asn1c/).

There are also many other ASN.1 compilers to choose from, depending on the target platform and target programming language. The ASN.1 specification for the ImageServer to ProcessServer is listed below.

```plaintext
--
-- ImageServer-ProcessServer INTERFACE SPECIFICATION
--
-- Notes
-- =====
-- All messages will be prefixed by a 32bit integer in network byte order
-- (most significant byte first). This prefix will contain the length (in bytes) of
-- the message (excluding the prefix). This prefix is not part of the ASN.1 message
-- definition!
-- This integer is needed because TCP/IP is stream based and not message based.
--
-- Sockets
-- ======
-- The ImageServer-ProcessServer interface will be implemented via a single TCP/IP
-- socket on each machine.
-- The ProcessServer system is the server, ImageServer is the client.
--
-- transaction-id
-- ============
-- Every message will have a transaction id. Responses to messages will use the
-- transaction id of the original message. Not every message will require a response.

-- If a message needs no response then a transaction id of zero may be used.
-- ProcessServer allocated transaction-id's will be positive whole numbers.
ImageServer
-- allocated transaction-id's will be negative. This will allow the recipient of
-- a message to determine whether a message is a response or an unsolicited
-- message.
--
-- Link Establishment
-- ==============
-- When the ImageServer connects to the ProcessServer, the ImageServer starts
-- by sending an alive message.
--
-- link-check Messages
-- =============
```
-- The link will be 'kept alive' by transmitting a 'keep alive' message
-- on the link periodically when there is no other traffic in that
-- direction on the link. The keep alive message will consist of 4
-- bytes containing zeros (i.e. a byte count indicating that a 0 length
-- message follows.)
-- This will be implemented by both the ProcessServer and ImageServer.
-- Both the ProcessServer and ImageServer will assume the link is dead
-- if no messages are received over the link for a period time.
--
-- There will be no error detection or correction over that provided by
-- TCP.
--
-- Message acknowledgement
-- ------------------------
-- If the ImageServer or the ProcessServer does not receive an expected
-- acknowledgement message within a specific time-out period, then the
-- ImageServer or the ProcessServer retransmits the initiating message.
-- The retransmitted message is an exact copy of the original message.
-- This restriction also applies to the transaction-id.
--
-- If the ImageServer does not receive the expected acknowledgement within the time-out
-- period, then the ImageServer or the ProcessServer continues to retransmit
-- the initiating message every time-out period.
-- It will do this for some specified number of times.
-- Receiving an acknowledgement during this time, “completes” the protocol
-- normally.
-- If the ImageServer hits the limit in the number of transmissions and does
-- not receive an acknowledgement, then the ImageServer closes the socket,
-- and connects again, and starts the protocol from "alive"
--
--
-- MESSAGES SENT BY ImageServer
-- MESSAGE                RESPONSE              COMMENT
-- alive                  (init)
-- vdt                    vdt-ack
-- error                  error-ack
--
-- MESSAGES SENT BY ProcessServer
-- MESSAGE                RESPONSE              COMMENT
-- init                   init-ack
-- send-file-path         file-path
-- del-data               del-data-ack
-- status-inq             status
-- config                 status
--
--

-- Modification History:
-- Date        Author                   Version     Description
-- 22-09-2005  Bent Nicolaisen, JAI     3           Added new Errorconditions
-- 19-07-2005  Bent Nicolaisen, JAI     2           Beta version made together with CSC
-- 24-06-2005  Bent Nicolaisen, JAI     1           Alpha version send to CSC
-- 01-06-2005  Bent Nicolaisen, JAI     0           Initial version
--

ImageSrvProcessSrv-Module
DEFINITIONS AUTOMATIC TAGS ::= BEGIN

--*****************************************************************************
--                   DEFINITION OVERVIEW
--*****************************************************************************

-- This file contains the definition of the interface between the ImageServer
-- and the ProcessServer
--
--
maxint INTEGER ::= 2147483647

maxcamera INTEGER ::= 16

TransactionId ::= INTEGER(-2147483647..2147483647)

UnicodeString ::= UTF8String

Message ::= SEQUENCE {
  transaction-id TransactionId,
  body MessageBody
}

MessageBody ::= CHOICE {
  alive          Alive, -- Issued by the ImageServer at the end of initialization
                  -- to establish the existence of the ImageServer with
                  -- the ProcessServer, and to begin the initialization
                  -- protocol.
  vdt            Vdt, -- Informs the ProcessServer that vehicle data is ready.
                  -- The message contains all the lanecontroller event
-- information
-- (time, lane number etc.) and the
-- vehicle description number to access the information
-- and the image file numbers to request the vehicle
-- image data.

vdt-ack     VdtAck,  -- Acknowledges the ImageServer’s VDT message.
error       Error,   -- Informs the ProcessServer of a change in
-- subsystem states.
error-ack   NULL,    -- Acknowledges the ImageServer’s ERROR message.

init        Config,  -- Instructs the ImageServer to initialize to the
-- specified parameters.
init-ack    NULL,    -- Acknowledges the ProcessServer’s INIT message.
file-path-ing FilePathInq, -- Instructs the ImageServer to send a specific
-- vehicle file path to the ProcessServer.
file-path   FilePath, -- Returns the vehicle file path requested by
-- the ProcessServer.
del-data    DelData, -- Instructs the ImageServer to delete specific range of
-- vehicle data files.
status-inq  NULL,    -- Instructs the ImageServer to send status.
status      Status,  -- Informs the ProcessServer of the ImageServer’s current
-- state and parameters.
config      Config,  -- Instructs the ImageServer to configure its state or
-- specified parameters.

...  -- additional messages will be silently ignored by earlier versions

Alive ::= SEQUENCE {
server-id       ServerID
}

Vdt ::= SEQUENCE {
vdt-number      VdtId,
server-id       ServerID,
site-id         SiteID,
lane-number      LaneID,
image-status     SEQUENCE SIZE(1..6) OF ImageStatus,
vdt-status       VdtStatus,
lane-controller-data LaneControllerData OPTIONAL,
match-data       MatchData OPTIONAL,
program-version  ProgramVersion,
date-time        GeneralizedTime,
img-srv-checksum CheckSum
}
VdtAck ::= SEQUENCE {
    vdt-number VdtId
}

FilePathInq ::= SEQUENCE {
    file-id FileId,
    file-type FileType
}

FilePath ::= SEQUENCE {
    file-id FileId,
    file-type FileType,
    file-path UnicodeString
}

DelData ::= SEQUENCE {
    upper-bound-vdt-number VdtId
}

Status ::= SEQUENCE {
    config-status Config, -- the read-write status
    error Error, -- last error
    vdt-number VdtId, -- last vdt generated
    number-of-images Counter, -- total number of images received this "run"
    number-of-vdt Counter, -- total number of VDT's generated this "run"
    number-of-lc-msg Counter, -- total number of Lane Controller
    -- received this "run"
    camera-stat SEQUENCE SIZE (0..maxcamera) OF CameraStatistics,
    program-version ProgramVersion, -- Program version of ImgSrv
    date-time GeneralizedTime, -- current time on ImgSrv
    img-srv-checksum CheckSum -- Checksum of ImgSrv program
}

Config ::= SEQUENCE {
    transmit-number VdtId OPTIONAL,
    image-server-state ServerState OPTIONAL,
    transmit-mode TransmitMode OPTIONAL,
    idle-limit IdleLimit OPTIONAL,
    message-timeout MessageTimeout OPTIONAL,
    data-collection SEQUENCE SIZE (1..maxcamera) OF DataCollection OPTIONAL
}

DataCollection ::= SEQUENCE {
    site-id SiteID, -- Site identifier
    lane-number LaneID, -- Lane identifier
    data-collect BOOLEAN -- Collect data on this lane?
}
Interfaces

ImageStatus ::= SEQUENCE {
  file-id FileId,
  camera-id CameraId
}

VdtStatus ::= SEQUENCE {
  missing-lanecontroller-data BOOLEAN, -- we got no lanecontroller data for this vdt
  plate-finder-status PlateFinderStatus, -- the status from the plate finder
  plate-type-found PlateType, -- the type of plate found
  vdt-file-status VdtFileStatus -- The status of the VDT file on the ImageServer
}

PlateFinderStatus ::= ENUMERATED { no-plate-finder-status, plate-found, plate-not-
found, plate-finder-error, image-error}
PlateType ::= ENUMERATED { no-plate-type, rectangular-plate, square-plate,
motorcycle-plate }
VdtFileStatus ::= ENUMERATED { no-vdt-file-status, file-ok, file-not-found, error-
in-file, ... }

LaneControllerData ::= SEQUENCE {
  date-time GeneralizedTime, -- Date and time from the lanecontroller
  site-id SiteID, -- Site identifier
  lane-number LaneID, -- Lane identifier
  data UnicodeString -- The lanecontroller data
}

KeyId ::= INTEGER(1..9999999) -- A unique number identifying the image
CameraId ::= INTEGER
VdtId ::= INTEGER(1..9999999) -- A unique number identifying the image
LaneId ::= INTEGER

MatchData ::= OCTET STRING

Error ::= SEQUENCE {
  server-id ServerID, 
  site-id SiteID OPTIONAL,
  lane-number LaneID OPTIONAL,
  camera-id CameraId OPTIONAL,
  device ErrorDevice,
  device-error INTEGER, -- Error code from device
  condition ErrorCondition,
  date-time GeneralizedTime
}
ErrorDevice ::= ENUMERATED {
  no-error-device,
  disk-drive,
  clock,
  camera,
  lane-controller,
  light-sensor, -- ?????
  msg-queue,
  vdt-queue,
  camera-queue,
  finder,
  protocol,
  camera-error, -- Error direct from camera, look in device-error for error code
...
}

ErrorCondition ::= ENUMERATED {
  no-error-condition,
  one-time-error, -- error event, will not be cleared later!
  inoperable, -- not working
  operable, -- OK
  over-80-percent-full, -- disk 80% full
  full, -- disk 100% full
  invalid-protocol, -- protocol error
  bad-delete-msg-parameters, -- bad parameters in msg
  bad-vdt-ack-msg-parameters, -- bad parameters in msg
  bad-msg-data-length, -- bad length of message
  no-message, -- no message, e.g from lane-controller
  extra-message, -- extra message, e.g. from lane-controller
...
}

CameraStatistics ::= SEQUENCE {
  site-id               SiteID,
  lane-number           LaneID,
  camera-id             CameraId,
  number-of-images      Counter,      -- Number of images received
  number-of-lc-msg      Counter,      -- Number of lanecontroller msg received
  number-of-flash       Counter,      -- Number of flash fired

  -- many other counters from the camera, TBD
...
}

IdleLimit ::= INTEGER (1..3600)
MessageTimeout ::= INTEGER (1..3600)
7.2 Lane Controller interface

- The ImageServer has a listen socket that listens for connections from lane controllers.
- The interface uses clean ASCII text lines. One message for each line of text.
- The ImageServer does not send anything to the lane controller.
- The messages that the lane controller can send to the ImageServer are:

**Keep alive:**
- Messagetype;
- Where Messagetype is “0”
- Sample message “0;”

**Trigger information:**
- Messagetype; Date; Time; SiteID; LaneNumber; MessageText
- Where Messagetype is “1”
- Sample message “1;2006-07-06;13:24:03.505;0;0;Lane controller message text”

**Error information:**
- Messagetype; Date; Time; SiteID; LaneNumber; ErrorNumber; ErrorMessageText
- Where Messagetype is “2”
- Sample message “2;2006-07-06;15:26:47.291;0;0;1;This is the error text;”
Based on the Site ID and the lane number the trigger information is placed in the queue for that site/lane. The date time will is used to correlate the trigger information with the images received for the site/lane.

The error information is relayed to the Process Server, but it is not processed in the ImageServer.
1 Appendix A: PC Minimum Specifications

Software:
- Windows XP/Server 2003
- .NET framework 1.1 and 2.0

Hardware:
- Pentium 4, 3GHz stationary PC equivalent, or faster
- 512 MB RAM
- Preferable Gigabit LAN support