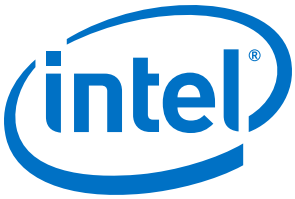

OpenCL* on Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA Quick Start User Guide

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **1.0 Production**

UG-20106 | 2018.04.11
Latest document on the web: PDF | HTML

Contents

1. About this Document

1.1. Conventions

Table 1. Document Conventions

Convention	Description
#	Precedes a command that indicates the command is to be entered as root.
$	Indicates a command is to be entered as a user.
`This font`	Filenames, commands, and keywords are printed in this font. Long command lines are printed in this font. Although long command lines may wrap to the next line, the return is not part of the command; do not press enter.
<variable_name>	Indicates the placeholder text that appears between the angle brackets must be replaced with an appropriate value. Do not enter the angle brackets.

1.2. Acceleration Glossary

Table 2. Acceleration Stack for Intel® Xeon® CPU with FPGAs Glossary

Term	Abbreviation	Description
Intel® Acceleration Stack for Intel Xeon® CPU with FPGAs	Acceleration Stack	A collection of software, firmware and tools that provides performance-optimized connectivity between an Intel FPGA and an Intel Xeon processor.
Intel Programmable Acceleration Card with Intel Arria® 10 GX FPGA	Intel PAC with Intel Arria 10 GX FPGA	PCIe* accelerator card with an Intel Arria 10 FPGA. Programmable Acceleration Card is abbreviated PAC. Contains a FPGA Interface Manager (FIM) that connects to an Intel Xeon processor over PCIe bus.
Intel Xeon Scalable Platform with Integrated FPGA	Integrated FPGA Platform	A platform with the Intel Xeon and FPGA in a single package and sharing a coherent view of memory using the Intel Ultra Path Interconnect (UPI).

1.3. Acronyms

Table 3. Acronyms

Acronyms	Expansion	Description
AFU	Accelerator Functional Unit	Hardware Accelerator implemented in FPGA logic which offloads a computational operation for an application from the CPU to improve performance.
AF	Accelerator Function	Compiled Hardware Accelerator image implemented in FPGA logic that accelerates an application.
API	Application Programming Interface	A set of subroutine definitions, protocols, and tools for building software applications.
FIM	FPGA Interface Manager	The FPGA hardware containing the FPGA Interface Unit (FIU) and external interfaces for memory, networking, etc. The Accelerator Function (AF) interfaces with the FIM at run time.
OPAE	Open Programmable Acceleration Engine	The OPAE is a software framework for managing and accessing AFs.

2. Introduction

This user guide describes how to get started with the OpenCL* on the Intel PAC with Intel Arria 10 GX FPGA 1.0 Production Release. The instructions use the precompiled OpenCL kernels included in this 1.0 Production Release. This user guide also includes a brief introduction to compiling OpenCL kernels.

OpenCL designs comprises two components, the kernel and the host. The kernel includes the accelerator code. The host runs on the host machine. The accelerator card plugs into the host machine.

Note: You must have root permission on the host machine to setup OpenCL.

Related Information

- Intel FPGA SDK for OpenCL - Getting Started Guide
  For more information about installation of the Intel FPGA Software Development Kit (SDK) for OpenCL and instructions on how to compile an example OpenCL application.
- Intel FPGA SDK for Open Computing Language (OpenCL) web-page
  For more details about OpenCL

2.1. Release Content

The archive file (`.tar.gz`), includes the files for the Intel PAC with Intel Arria 10 GX FPGA Production Release. The release includes the following files for OpenCL located in the `<installation_path>/opencl` folder:

- 1.0 OpenCL Board Support Package (BSP):
  - `opencl_bsp.tar.gz`
- OpenCL example designs tested with :
  - `exm_opencl_hello_world_x64_linux.tgz`
  - `exm_opencl_vector_add_x64_linux.tgz`
- Pre-compiled kernels <aocx>:
  - `hello_world.aocx`
  - `vector_add.aocx`

Related Information

Extracting the Intel PAC with Intel Arria 10 Package

3. Setting Up the Host Machine

Prerequisites: You must follow the instructions from the *System Requirements* and *Installing Required OS Packages and Components While Installing CentOS 7.4* sections of the *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Arria 10 GX FPGA*, referred to as *Quick Start Guide* throughout this document before you start setting up the host machine.

Attention:

- If you require the OpenCL compiler and tools to build and run OpenCL applications, download and install the Intel FPGA SDK for OpenCL.
- If you only require the Intel FPGA SDK for OpenCL's kernel deployment functionality, download and install the Intel FPGA RTE for OpenCL.
- Do not install the RTE and the SDK on the same host system. The SDK already contains the RTE.

Related Information

- Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA
- System Requirements
- Installing Required OS Package and Components While Installing CentOS 7.4

3.1. Installing the Intel FPGA RTE for OpenCL

Follow the instructions from the *Installing the Intel Acceleration Stack Runtime package on the Host Machine* section of the *Quick Start Guide* to install the Intel Acceleration Stack Runtime package.

Depending on your selection, the OpenCL Runtime Environment (RTE) is installed in one of the following install directories:

- Acceleration Stack for Runtime:

```
/home/<username>/intelrtestack
```

- Custom install directoy:

```
/<custom_install_directory>
```

The above paths, referred to as *<RTE install path>* throughout this document.

Related Information

Installing the Intel Acceleration Stack Runtime Package on the Host Machine

3.2. Installing the Intel FPGA SDK for OpenCL

Follow the instructions from the *Installing the Intel Acceleration Stack Development Package on the Host Machine* section of the *Quick Start Guide* to install the Acceleration Stack Development package.

Depending on your selection, the OpenCL SDK is installed in one of the following install directories:

- Acceleration Stack for Development:

```
/home/<username>/inteldevstack
```

- Custom install directoy:

```
/<custom_install_directory>
```

The above paths, referred to as *<OpenCL SDK install path>* throughout this document.

Related Information

Installing the Intel Acceleration Stack Development Package on the Host Machine

3.3. Installing the Release

You build the OpenCL BSP provided with this release on top of the Intel Acceleration Stack for Intel Xeon CPU with FPGAs. Follow the entire *Quick Start Guide* to set up the Intel PAC with Intel Arria 10 GX FPGA.

Note: Ensure that you have installed the Intel PAC with Intel Arria 10 GX FPGA, updated the BMC firmware, flashed the FPGA with the FIM image, and installed the OPAE package.

You can run the OPAE software in a non-virtualized environment or in a virtualized environment with Single Root IO Virtualization (SR-IOV) disabled.

To run the OPAE in a virtualized environment that includes virtual functions (VFs) and SR-IOV, complete the following additional steps:

1. For OpenCL functionality, load the OpenCL configuration before enabling SR-IOV mode.
2. Set the `CL_CONTEXT_COMPILER_MODE_ALTERA` environment variable to disable FPGA configuration or reconfiguration during OpenCL host runtime:

```
$ export CL_CONTEXT_COMPILER_MODE_ALTERA=3
```

After completing these steps, the following environment variable is available:

- `DCP_LOC` – points to the location of the extracted release archive.

Note: To avoid having to reset `DCP_LOC` after a reboot, save this variable to your shell initialization script.

Related Information

Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

3.4. Installing the OpenCL BSP

The OpenCL BSP is an archive file. To extract the OpenCL BSP directory, type the following commands:

```
cd $DCP_LOC/opencl/

$ tar xf opencl_bsp*.tar.gz

$ cd opencl_bsp

$ export AOCL_BOARD_PACKAGE_ROOT=`pwd`
```

To avoid having to reset the AOCL_BOARD_PACKAGE_ROOT environment variable after a reboot, save it to your shell initialization script.

3.5. Setting Up Permissions

Running OpenCL requires you to set various permissions and system parameters. Running the `setup_permissions.sh` script completes this task. The script uses `sudo` internally; consequently, it requires root privileges.

Procedure:

1. Run the script once, when you enable OpenCL for first time on this host:

   ```
   $ $AOCL_BOARD_PACKAGE_ROOT/linux64/libexec/setup_permissions.sh
   ```

   Note: The script requires `sudo` access. You need to enter password for that.
2. Reboot the computer because some permanent settings only take effect after a reboot.
3. Some of the settings are not permanent. Consequently, you must rerun the `setup_permissions.sh` command after rebooting.

   ```
   $ $AOCL_BOARD_PACKAGE_ROOT/linux64/libexec/setup_permissions.sh
   ```

3.6. Initializing RTE

Before running OpenCL examples, you must initialize the RTE.

In addition to the previously mentioned variables, set one of the following additional environment variables for the RTE:

- If you have installed Intel Acceleration Stack Runtime Package, set the following:

```
export ALTERAOCLSDKROOT=/<RTE install path>/intelFPGA_pro/aclrte-linux64
```

- If you have installed Intel Acceleration Stack Development Package, set the following:

```
export ALTERAOCLSDKROOT=/<OpenCL SDK install path>/intelFPGA_pro/hld
```

Run the OpenCL initialization script from the RTE:

```
$ source $ALTERAOCLSDKROOT/init_opencl.sh
```

3.7. Setup Summary

Each time you reboot the computer, you must complete the following steps to run the OpenCL examples:

- Set the following environment variables:
  - DCP_LOC
  - AOCL_BOARD_PACKAGE_ROOT
- Run the permissions script:

```
$ $AOCL_BOARD_PACKAGE_ROOT/linux64/libexec/setup_permissions.sh
```

- Initialize RTE:

```
$ source $ALTERAOCLSDKROOT/init_opencl.sh
```

4. Running Diagnostics

Before running diagnostics, load an OpenCL kernel to the board. The following instructions use the `hello_world` kernel or you may also use your own.

1. Load `hello_world` OpenCL kernel:

```
$ aocl program acl0  $DCP_LOC/opencl/hello_world.aocx
```

   Sample program output:

```
aocl program: Running program from $DCP_LOC/opencl/opencl_bsp/linux64/
libexec
Program succeed.
```

2. Run the simple diagnostic utility:

```
$ aocl diagnose
```

   Sample diagnostic output:

```
aocl diagnose: Running diagnose from $DCP_LOC/opencl/opencl_bsp/linux64/
libexec

------------------------- acl0 -------------------------
Vendor: Intel Corp

Phys Dev Name  Status   Information

pac_a10_f400000          Passed   PAC Arria 10 Platform (pac_a10_f200000)
                            PCIe 04:00.0
                            FPGA temperature = 47 degrees C.

DIAGNOSTIC_PASSED
---------------------------------------------------------
```

3. Run the advanced diagnostic:

```
$ aocl diagnose acl0
```

   Sample advanced diagnostic output:

```
aocl diagnose: Running diagnose from <installation_path>/opencl/opencl_bsp/
linux64/libexec
Using platform: Intel(R) FPGA SDK for OpenCL(TM)
Using Device with name: pac_a10 : PAC Arria 10 Platform (pac_a10_f200000)
Using Device from vendor: Intel Corp clGetDeviceInfo
CL_DEVICE_GLOBAL_MEM_SIZE = 8589934592
clGetDeviceInfo CL_DEVICE_MAX_MEM_ALLOC_SIZE = 8588886016
Memory consumed for internal use = 1048576
Actual maximum buffer size 8588886016 bytes
Writing 8191 MB to global memory...
Allocated 1073741824 Bytes host buffer for large transfers
Write speed: 5447.76 MB/s [5100.38 -> 5710.86]
Reading and verifying 8191 MB from global memory ...
Read speed: 6319.11 MB/s [5829.62 -> 6815.82]
Successfully wrote and readback 8191 MB buffer
```

```
Transferring 262144 KBs in 512 512 KB blocks ... 3295.09 MB/s
Transferring 262144 KBs in 256 1024 KB blocks ... 3465.62 MB/s
Transferring 262144 KBs in 128 2048 KB blocks ... 4173.86 MB/s
Transferring 262144 KBs in 64 4096 KB blocks ... 5069.94 MB/s
Transferring 262144 KBs in 32 8192 KB blocks ... 5084.80 MB/s
Transferring 262144 KBs in 16 16384 KB blocks ... 5538.76 MB/s
Transferring 262144 KBs in 8 32768 KB blocks ... 6165.23 MB/s
Transferring 262144 KBs in 4 65536 KB blocks ... 6536.86 MB/s
Transferring 262144 KBs in 2 131072 KB blocks ... 6320.60 MB/s
Transferring 262144 KBs in 1 262144 KB blocks ... 6619.78 MB/s

As a reference:
PCIe Gen1 peak speed: 250MB/s/lane
PCIe Gen2 peak speed: 500MB/s/lane
PCIe Gen3 peak speed: 985MB/s/lane

Writing 262144 KBs with block size (in bytes) below:

Block_Size Avg    Max    Min    End-End (MB/s)
  524288 2509.11 3295.09 1693.93 2018.67
 1048576 2543.70 3087.25 1656.82 2279.26
 2097152 3634.87 4173.86 2265.05 3410.79
 4194304 4548.67 5069.94 3939.32 4362.32
 8388608 4813.88 5084.80 4089.09 4722.04
16777216 5266.92 5446.97 4821.61 5206.11
33554432 4818.27 5226.23 3681.99 4792.34
67108864 4964.35 5662.74 4123.11 4952.34
134217728 4367.72 4640.88 4124.93 4366.66
268435456 4546.45 4546.45 4546.45 4546.45

Reading 262144 KBs with block size (in bytes) below:

Block_Size Avg    Max    Min    End-End (MB/s)
  524288 2487.06 3038.19 1757.40 2015.28
 1048576 2934.13 3465.62 2241.64 2613.45
 2097152 3485.74 3673.13 2820.99 3296.42
 4194304 3406.50 3629.74 3040.80 3300.23
 8388608 4474.60 4589.06 4241.70 4378.74
16777216 5289.71 5538.76 5081.67 5219.55
33554432 6014.68 6165.23 5686.37 5976.21
67108864 6440.31 6536.86 6365.68 6421.60
134217728 6106.75 6320.60 5906.89 6098.65
268435456 6691.78 6691.78 6691.78 6691.78

Write top speed = 5662.74 MB/s
Read top speed = 6691.78 MB/s
Throughput = 6177.26 MB/s

DIAGNOSTIC_PASSED
```

5. OpenCL Support for Multi-Card Systems

Before running an OpenCL application, program the PAC card with an Accelerator Function (AF) that includes the BSP logic. Use the `aocl` program command to load an `aocx` file to the PAC card. It is only necessary to program the AF one time per PAC card. After the initial programming, you can use the OpenCL API to load different applications to the PAC card using the `aocx` program command.

Note: For a system with one PAC card, Intel recommends that you allocate the number of hugepages to 20. If your system has multiple PAC cards, you must allocate 20 hugepages per card. For an example, system with four PAC cards requires total 80 hugepages.

Run the `aocl diagnose -probe` command to determine how many FPGAs the system includes. For example, running the `aocl diagnose -probe` command on a system with three PAC cards might show output similar to the following:

1. `$ aocl diagnose -probe`

```
aocl diagnose: Running diagnose from $DCP_LOC/opencl/
boardtest/opencl_bsp_build/linux64/libexec
pac_a10_f200001
pac_a10_f200000
pac_a10_f200002
```

2. The following command programs the first card listed in Step 1:

   `$ aocl program pac_a10_f200001 $DCP_LOC/opencl/ hello_world.aocx`

```
aocl program: Running program from $DCP_LOC/opencl/
boardtest/opencl_bsp_build/linux64/libexec

Program succeed.
```

3. The following command programs the second card listed in Step 1:

   `$ aocl program pac_a10_f200000 $DCP_LOC/opencl/ hello_world.aocx`

```
aocl program: Running program from $DCP_LOC/opencl/
boardtest/opencl_bsp_build/linux64/libexec

Program succeed.
```

4. After programming the FPGAs, the `aocl diagnose` command provides information about them:

   `$ aocl diagnose`

```
aocl diagnose: Running diagnose from $DCP_LOC/opencl/
boardtest/opencl_bsp_build/linux64/libexec

------------------------ acl0 -------------------------
```

```
Vendor: Intel Corp

Phys Dev Name  Status   Information

pac_a10_f200001          Passed   PAC Arria 10 Platform (pac_a10_f200001)
                            PCIe 05:00.0
                            FPGA temperature = 50 degrees C.

DIAGNOSTIC_PASSED
---------------------------------------------------------


------------------------- acl1 ------------------------
Vendor: Intel Corp

Phys Dev Name  Status   Information

pac_a10_f200000          Passed   PAC Arria 10 Platform (pac_a10_f200000)
                            PCIe 03:00.0
                            FPGA temperature = 50 degrees C.

DIAGNOSTIC_PASSED
---------------------------------------------------------
```

6. Running Samples

This section describes how to compile and run the host code for the provided samples using the precompiled OpenCL kernels.

6.1. Running Hello World

1. Extract `hello_world` example:

```
$ cd $DCP_LOC/opencl
$ mkdir exm_opencl_hello_world_x64_linux
$ cd exm_opencl_hello_world_x64_linux
$ tar xf ../exm_opencl_hello_world_x64_linux.tgz
```

2. Build example:

```
$ cd hello_world
$ make
```

3. Copy `aocx` to example bin folder:

```
$ cp $DCP_LOC/opencl/hello_world.aocx ./bin/
```

4. Run example:

```
$ ./bin/host
```

Example sample output:

```
Querying platform for
info:

==========================
CL_PLATFORM_NAME                         = Intel(R) FPGA SDK for OpenCL(TM)
CL_PLATFORM_VENDOR                       = Intel(R) Corporation
CL_PLATFORM_VERSION                      = OpenCL 1.0 Intel(R) FPGA SDK
for OpenCL(TM), Version 17.0

Querying device for info:
========================
CL_DEVICE_NAME                           = pac_a10 : PAC Arria 10 Platform
(pac_a10_f400000)
CL_DEVICE_VENDOR                         = Intel Corp
CL_DEVICE_VENDOR_ID                      = 4466
CL_DEVICE_VERSION                        = OpenCL 1.0 Intel(R) FPGA SDK
for OpenCL(TM), Version 17.0
CL_DRIVER_VERSION                        = 17.0
CL_DEVICE_ADDRESS_BITS                   = 64
CL_DEVICE_AVAILABLE                      = true
CL_DEVICE_ENDIAN_LITTLE                  = true
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE          = 32768
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE      = 0
CL_DEVICE_GLOBAL_MEM_SIZE                = 8589934592
CL_DEVICE_IMAGE_SUPPORT                  = true
CL_DEVICE_LOCAL_MEM_SIZE                 = 16384
CL_DEVICE_MAX_CLOCK_FREQUENCY            = 1000
```

```
CL_DEVICE_MAX_COMPUTE_UNITS              = 1
CL_DEVICE_MAX_CONSTANT_ARGS              = 8
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE       = 2147483648
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS       = 3
CL_DEVICE_MEM_BASE_ADDR_ALIGN            = 8192
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE       = 1024
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR    = 4
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT   = 2
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT     = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG    = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT   = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE  = 0
Command queue out of order?              = false
Command queue profiling enabled?         = true
Using AOCX: hello_world.aocx
Reprogramming device [0] with handle 1

Kernel initialization is complete.
Launching the kernel...

Thread #2: Hello from Altera's OpenCL Compiler!

Kernel execution is complete.
```

6.2. Running Vector Add

1. Extract example:

```
$ cd $DCP_LOC/opencl
$ mkdir exm_opencl_vector_add_x64_linux
$ cd exm_opencl_vector_add_x64_linux
$ tar xf ../exm_opencl_vector_add_x64_linux.tgz
```

2. Build example:

```
$ cd vector_add
$ make
```

3. Copy precompiled OpenCL kernel to bin folder:

```
$ cp $DCP_LOC/opencl/vector_add.aocx ./bin
```

4. Run example:

```
$ ./bin/host
```

   Example sample output:

```
Initializing OpenCL
Platform: Intel(R) FPGA SDK for OpenCL(TM)
Using 1 device(s)
  pac_a10 : PAC Arria 10 Platform (pac_a10_f200000)
Using AOCX: vector_add.aocx
Reprogramming device [0] with handle 1
Launching for device 0 (1000000 elements)

Time: 8.046 ms
Kernel time (device 0): 3.711 ms

Verification: PASS
```

7. Compiling OpenCL Kernels

Prerequisite: You must install the OpenCL SDK using the instructions in the *Installing the OpenCL SDK* section before you start compiling the OpenCL Kernels.

1. Set the user environment variable with the following:

```
$ export ALTERAOCLSDKROOT=/<OpenCL SDK install path>/hld

$ export AOCL_BOARD_PACKAGE_ROOT=$DCP_LOC/opencl/opencl_bsp

$ export PATH=$ALTERAOCLSDKROOT/bin:$PATH

$ export LD_LIBRARY_PATH=$ALTERAOCLSDKROOT/host/linux64/lib:\
$LD_LIBRARY_PATH
```

2. Ensure that the environment is setup with correct BSP using the following command:

```
aoc --list-boards

Output
Board list:
pac_a10
```

3. Compile an OpenCL Kernel to an `aocx` using commands similar to the following:

```
cd $DCP_LOC/opencl/exm_opencl_vector_add_x64_linux/vector_add

$aoc device/vector_add.cl -o bin/vector_add.aocx --board pac_a10
```

You may get warning messages during the compilation of the OpenCL kernels. You can safely ignore these messages.

For additional information on Intel FPGA SDK for OpenCL, refer to the *Intel FPGA SDK for OpenCL Quick Start Guide*.

Related Information

- Intel FPGA SDK for OpenCL - Getting Started Guide
  For more information about installation of the Intel FPGA Software Development Kit (SDK) for OpenCL and instructions on how to compile an example OpenCL application.
- Installing the Intel FPGA SDK for OpenCL on page 7

7.1. Checking Timing Results

Intel recommends that you check for timing failures after compilation of the `aocx` file.

Check the compilation directory for the presence of the following report files:

```
afu_fit.failing_clocks.rpt
```

```
afu_fit.failing_paths.rpt
```

For example, after compiling `vector_add.cl`, locate the `$DCP_LOC/opencl/exm_opencl_vector_add_x64_linux/vector_add/device/vector_add` directory. If there is a timing violation, this directory contains the failing report files. The failing report files indicate that the the timing is not clean and the functional correctness cannot be guaranteed.

If OpenCL kernel compilation results in timing violations, Intel recommends to retry compilation with a different seed (`aoc <kernel.cl>--seed <integer>`).

For example,

```
aoc vector_add.cl --seed 2
aoc vector_add.cl --seed 3
aoc vector_add.cl --seed 63
```

8. Document Revision History for OpenCL on Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Quick Start User Guide

Document Version	Intel Acceleration Stack Version	Changes
2018.04.11	1.0 Production (supported with Intel Quartus® Prime Pro Edition 17.0.0)	• Made the following changes in *Release Content* section: — Updated the location of the OpenCL related files. — Corrected the file name for OpenCL BSP. • Updated the *Setting up the Host Machine* section to include the instructions about how to install the OpenCL RTE and OpenCL SDK. • Updated sample advanced diagnostic output in *Running Diagnostics* section. • Made the following changes in *OpenCL Support for Multi-Card Systems* section: — Added a note to clarify the allocation of the number of hugepages for a multicard system. — Modified the FPGA temperature for `pac_a10_f200001` and `pac_a10_f200000` cards. • Updated sample output in *Running Vector Add* section • Updated the compilation instructions in *Compiling the OpenCL Kernels* section. • Added new section *Checking Timing Results*.
2017.12.22	1.0 Beta (supported with Intel Quartus Prime Pro Edition 17.0.0)	Initial release.