RTX64

Features by Release

This document outlines support for key RTX64 features by product version. Additionally, it outlines compatibility between supported RTX64 versions, Windows operating system versions and recommended *tested* Service Pack combinations and supported versions of Microsoft Visual Studio.

Previous and subsequent operating system Service Packs may work but have not been tested by IntervalZero, and therefore cannot be guaranteed to work. If in doubt about hardware or software requirements, please contact IntervalZero Support.

 IZ-DOC-X64-0242-R3

Operating System Compatibility

NOTE: Windows Home Editions are not tested.

Windows Version	RTX64 3.1	RTX64 3.2 w/ Updates	RTX64 3.3 w/ Updates	RTX64 3.4 w/ Updates	RTX64 3.5 w/ Updates	RTX64 3.6 w/ Updates	RTX64 3.7 w/ Updates	RTX64 4.0 w/ Updates	RTX64 4.1
10	Yes (Semi-Annual Channel 1607)	Yes (Semi-Annual Channel 1703, 1607)	Yes (Semi-Annual Channel 1703, 1607)	Yes (Semi-Annual Channel 1803, 1709, 1703, 1607)	Yes (Semi-Annual Channel 1809~, 1803, 1709, 1703, 1607)	Yes (Semi-Annual Channel 1809, 1803, 1709+)	Yes (Semi-Annual Channel 2009‡, 2004‡, 1909, 1903, 1809, 1803, 1709+)	Yes (Semi-Annual Channel 2009*, 2004*, 1909, 1903, 1809+)	Yes (Semi-Annual Channel 2009, 2004, 1909, 1903, 1809+)
10 IoT Enterprise	Yes (LTSB 1607)	Yes (LTSB 1607)	Yes (LTSB 1607)	Yes (LTSB 1607)	Yes (LTSB 1607)	Yes (LTSC 2019, LTSB 1607)	Yes (LTSC 2019, LTSB 1607)	Yes (LTSC 2019, LTSB 1607)	Yes (LTSC 2019, LTSB 1607)
	Please contact sales@intervalzero.com if you are interested in IoT Core.								
8.1	Yes (8. 1 with Update)	Yes (8. 1 with Update)	Yes (8. 1 with Update)	Yes (8. 1 with Update)	Yes (8. 1 with Update)	Yes (8. 1 with Update)	Yes (8. 1 with Update)	No	No
7 SP1	Yes	Yes	Yes	Yes	Yes^	Yes^	Yes^	No	No
WES 8	Yes (8.1 with Update)	Yes (8.1 with Update)	Yes (8.1 with Update)	Yes (8.1 with Update)	Yes (8.1 with Update)	Yes (8.1 with Update)	Yes (8.1 with Update)	No	No
WES 7	Yes (SP1)	Yes (SP1)	Yes (SP1)	Yes (SP1)	Yes (SP1)	Yes (SP1)	Yes (SP1)	No	No

~ Semi-Annual Channel 1809 should work with RTX64 3.5 but was not tested with that version.
+ Previous Windows 10 versions should work but were not tested for this release.
‡ Support for Windows 10 Feature Updates 2009 and 2004 was added in Update 3 for RTX64 3.7
* Support for Windows 10 Feature Updates 2009 and 2004 was added in Update 2 for RTX64 4.0
^ Windows Updates KB2921916 and KB3033929 are required for SHA-2 signing.

Operating System Compatibility (continued)

NOTE: Windows Home Editions are not tested.

Windows Version	RTX64 2014	RTX64 2014 w/ SP1	RTX64 2014 w/ SP2	RTX64 3.0 w/ Updates
10	No	No	No	Yes (Semi-Annual Channel 1607)
10 IoT Enterprise	No	No	No	No
8.1	No	No	Yes (8.1 with Update)	Yes (8.1 with Update)
7 SP1	Yes	Yes	Yes	Yes
WES 8	Yes	Yes	Yes (8.1 with Update)	Yes (8.1 with Update)
WES 7	Yes (SP1)	Yes (SP1)	Yes (SP1)	Yes (SP1)

Visual Studio Compatibility

Visual Studio Version	RTX64 3.1	RTX64 3.2 w/ Updates	RTX64 3.3 w/ Updates	RTX64 3.4 w/ Updates	RTX64 3.5 w/ Updates	RTX64 3.6 w/ Updates	RTX64 3.7 w/ Updates	RTX64 4.0 w/ Updates	RTX64 4.1
2019	No	No	No	No	No	No	Yes	Yes	Yes
							Enterprise, Professional, and Community editions supported		
2017	Yes (Update 2)	Yes (Update 2)	Yes (Update 3)	Yes (Update 3)	Yes (Update 7)	Yes (Update 8)	Yes (Update 9)	Yes (Update 9)	Yes (Update 9)
	Enterprise, Professional, and Community editions supported								
2015	Yes (Update 3)	Yes (Update 3)	Yes (Update 3)	Yes (Update 3)	Yes (Update 3)	Yes (Update 3)	Yes (Update 3)	*Deprecated. Support will be removed in the next major release.*	*Deprecated. Support will be removed in the next major release.*
	Ultimate, Premium, Pro, and Community editions supported								
2013	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	No	No
2012	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	*Deprecated.*	No	No

Visual Studio Compatibility (continued)

Visual Studio Version	RTX64 2014	RTX64 2014 w/ SP1	RTX64 2014 w/ SP2	RTX64 3.0 w/ Updates
2019	No	No	No	No
2017	No	No	No	No
2015	No	No	Yes (requires RTX64 Visual Studio 2015 support)	Yes (Update 2; Ultimate, Premium, Pro, Community editions supported)
2013	No	Yes	Yes	Yes (Update 3; Ultimate, Premium, Pro, Community editions supported)
2012	Yes (Building only, no debugging)	Yes	Yes	Yes (Update 1 or greater)

Support for Key Features

General / System

Feature	Supported in	Notes
Windows Groups Security	All	
Windows Secure Boot	RTX64 3.6 and later	
Dedicated Mode (up to 63 processors for RTX64)	All	
Deterministic memory allocation	All	Local memory was re-architected in RTX64 4.0
System Tray integration	RTX64 3.4 and later	

Subsystem Configuration and Optimization

Feature	Supported in	Notes
Activation and Configuration Utility	All	
Control Panel	All	
Managed Code Framework Interface to Configure the Subsystem (SDK)	All	RTX64 4.0 contains breaking API changes that may not be compatible with earlier binaries. See the Help for more information.
Native Framework Interface to Configure the Subsystem (SDK)	All	RTX64 4.0 contains breaking API changes that may not be compatible with earlier binaries. See the Help for more information.

Feature	Supported in	Notes
Performance Optimization with Intel® RDT	Update 2 for RTX64 3.3 and later	This functionality is hardware-dependent. Update 2 for RTX64 3.3 – RTX64 3.4 and updates: • CAT / MBA *Flat* mode only RTX64 3.5 and later: • Enable/ disable • CAT / MBA *Flat* or *Priority-based CLOS performance* modes

Application Development

Feature	Supported in	Notes
Application and RTDLL Templates	All	
Structured Exception Handling (SEH)	All	Enabled/disabled globally, not by feature, in RTX64 2014 with Service Pack 1 and later
Floating Point	All	
MMX, SSE/SSE2/SSE3/SSE4	All	
AVX	All	• RTX64 2014 – RTX64 3.3 supports version 2.0 • RTX64 3.4 and later supports version 512
Microsoft C Runtime	All	• Support for functions *errno, fwrite, fflush, ferror, feof,* and *clearer* was added in RTX64 3.1 • Support for library call *strftime* was added in RTX64 3.3

Feature	Supported in	Notes
		• Support for functions *assert* and *abort* was added in RTX64 3.5

Application Debugging

For the features below, support is dependent on Visual Studio version compatibility with RTX64. See *Visual Studio Compatibility* above.

Feature	Supported in	Notes
Local Debugging	All	
Remote Debugging	RTX64 2014 with Service Pack 1 and later	
Debugger Launch	All	
Debugger Local Attach	RTX64 3.1 and later	Support for local and running RTSS processes.
Debugger Remote Attach	RTX64 4.1 and later	

Tools & Utilities

Feature	Supported in	Notes
Tools to Start/Stop Processes	All	RtssRun, RtssKill, Task Manager

Feature	Supported in	Notes
Schedule Auto-start Processes	RTX64 3.5 and later	This functionality is available in Task Manager.
Console to Display Process Output	All	The console window, RTX64 Console, was re-architected in RTX64 4.1. Functionality was added to the Control Panel for configuring RTX64 Console and logging of real-time application output.
Tool to Evaluate Performance	All	This functionality is available in Latency View.
Tool to Trace Process Behavior	All	This functionality is available in Monitor.
Tool to Display Process Behavior	RTX64 3.1 and later	Available via text file (Runtime) and Tracealyzer for RTX64 (SDK)
Tool to Display CPU Usage	RTX2014 with Service Pack 2 and later	RTX2014 with Service Pack 2 – RTX64 3.4 • Command line only RTX64 3.5 and later • Functionality available in Task Manager
Tool to Display Object State (Command Line)	RTX2014 with Service Pack 2 and later	
Tool to Collect System Information (Command Line)	RTX2014 and later	This functionality is available in Analyzer.

Network Devices

Feature	Supported in	Notes
Plug and Play Devices	All	
Line-based Interrupts	All	
Message-based & Extended Message-based Interrupts	All	
Multiple Vector Interrupt Support for MSI-X	RTX64 3.3 and later	

Network Support

Feature	Supported in	Notes
Network Abstraction Layer (NAL)	RTX64 4.0 and later	The RT-TCP/IP Stack is layered on the NAL.
Timestamping	RTX64 4.0 and later	
IPv4	All	
IPv6	All	
UDP	All	
TCP	All	
ICMPv4	All	
ICMPv6	All	

Feature	Supported in	Notes
ARP	All	
Ethernet	All	
Multicast	All	
Raw Sockets	All	
Jumbo Packets	All	Support is driver-dependent.
MAC Layer Filter (Single Layer)	All	
Basic Winsock 2.0	All	
Multi-Homing	RTX64 3.2 and later	
Virtual Network (Single Network)	All	The Virtual Network was re-architected for RTX64 4.1
Network Tools	All	RtssArp, RtssIpConfig, RtssPing, RtssRoute

Key Features by Release

RTX64 4.1

General

- Added support for Windows 10 Update Version 2009 (20H2).
- Added support for Windows 10 Update Version 2004 (Windows 10 June 2020 Update). (9080)
- Improved the performance of RTSS-to-Windows communication on Windows 10 Feature Update Version 2004 from the initial implementation in the RTX64 4.0.1 patch. (9578)

Subsystem

- Implemented RTSS Timer Tick Compensation to account for SMI and other non-maskable system behaviors that impact latency and cause jitter. (8967)

Tools and Utilities

- Added RTX64 Console, which is an output display console window that displays output per real-time application.
- Added functionality to the Control Panel to support configuration of:
    - RTX64 Console (single or per process) and logging of real-time application output.
    - Remote debugging connections.
- Improved RTX64 Analyzer to display a more complete installation history. (7898)
- Added functionality to RtssRun and Task Manager that allows you to specify the time-of-allocation for the process external MSpace. (9640)
- Added the ability to reset default values for the memory profile settings *Available system memory* and *Percentage expected to be used by Windows*. (8844)

Application Debugging

- Added support for attaching the Visual Studio debugger to an RTSS process running on a remote system, outside of the Visual Studio IDE. See the SDK Help for more information.

Real-Time APIs

- Added new Real-time APIs that retrieve shared memory information:
  - RtQuerySharedMemory retrieves shared memory information
  - Structure SHARED_MEMORY_INFO contains shared memory information retrieved by RtQuerySharedMemory.
- Added attribute value RT_PROC_THREAD_ATTRIBUTE_ALLOC_EXTERNAL_MSPACE_AT_PROCESS_START to Real-time function RtUpdateProcThreadAttribute. This value specifies the time-of-allocation for the process's external MSpace. (9635)
- Added logic to Real-time function RtTerminateProcess so that it now fails and sets last error to ERROR_INVALID_HANDLE when an invalid handle is specified for parameter hProcess. (8000)

Native Framework APIs

- Added new Real-time Native Framework functions and structures for configuring real-time application output consoles (9239)
  - RtfwSetConsoleConfigurationEx configures real-time application output consoles.
  - RtfwGetConsoleConfigurationEx retrieves the configuration of real-time application output consoles.
  - Structure RTFW_CONSOLE_CONFIGURATION_EX represents the configuration of real-time application output consoles.
- Added new Real-time Native Framework functions and structures for configuring remote debugging:
  - RtfwGetRemoteDebuggerConfiguration retrieves the RTX64 remote debugger configuration.
  - RtfwSetRemoteDebuggerConfiguration sets the RTX64 remote debugger configuration.
  - Structure RTFW_REMOTE_DEBUGGER_CONFIGURATION represents the configuration of the RTX64 remote debugger.
- Added new member AllocateExtMSpaceAtStartup to Native Framework structure RTFW_SCHEDULED_PROCESS that specifies whether the process will allocate the process external MSpace at process start. (9224)

Managed Framework APIs

- Added new properties and methods to Managed Framework class ServerConsole (9239):
  - *AutoClose* gets/sets whether real-time application output consoles close automatically. If true, the consoles close automatically, otherwise they remain open until the user closes them. When configured to close automatically, and if property UsePerProcessConsoles is true, each console closes when its corresponding real-time process terminates. Otherwise, the single console closes when the Subsystem stops.
  - *LogFolder* gets/sets the pathname of the folder where real-time application output console log files are written. If this pathname doesn't exist, an exception is thrown.

- *LogFolderMaxDiskSize* gets/sets the maximum size of the real-time application output console log folder, in megabytes. The minimum valid value is 1 MB. The maximum valid value is uint.MaxValue MB.
- *TCPPort* gets/sets the TCP port number on which RTX64 Server listens for real-time application output console connections. The minimum valid value is 1. The maximum valid value is 65535.
- *UsePerProcessConsoles* gets/sets whether or not each real-time process displays output in its own real-time application output console. If true, per-process consoles are used, otherwise only one console shows output from all real-time processes.
- *ResetAutoClose* resets property AutoClose to its default value.
- *ResetLogFolder* resets property LogFolder to its default value.
- *ResetLogFolderMaxDiskSize* resets property LogFolderMaxDiskSize to its default value.
- *ResetTCPPort* resets property TCPPort to its default value.
- *ResetUsePerProcessConsoles* resets property UsePerProcessConsoles to its default value.

- Added new properties and methods to Managed Framework class IntervalZero.RTX64.Config.RemoteDebugger for configuring remote debugging:
  - *RemoteDebuggerEnable* determines whether remote debug attach connections to the computer are allowed.
  - *RemoteDebuggerName* gets/sets the IPv4 address of the host system.
  - *RemoteDebuggerPort* gets/sets the Port ID of the target system. This value must be between 1 and 65534. The default is 31094.
  - *RemoteDebuggerUriInUse* retrieves the Uniform Resource Identifier (URI) in use by the target system.
  - *Reset* resets all remote debugger properties to their default values.
  - *ResetEnable* resets the RemoteDebuggerEnable property to its default value (disabled).
  - *ResetName* resets the RemoteDebuggerName property to the default IPv4 address of the host system.
  - *ResetPort* resets the RemoteDebuggerPort property to the default Port number 31094.
- Added new property AllocateProcessExtMSpaceAtStartup to Managed Framework class IntervalZero.RTX64.Config.ScheduledProcess that specifies whether the process will allocate the process external MSpace at process start. (9224)
- Added new overload Start(Dictionary) to the Managed Framework RTProcess.Start Method API which starts an RTSS process via a dictionary that uses the StartParameters enumeration as the dictionary's key type. (9224)

RTX64 4.0

Key Features

Memory

RTX64 4.0 implements a completely new local memory allocation architecture. The local memory configuration now contains multiple allocation spaces. See Help topic *Allocation Spaces in Local Memory* for more information. (4509)

Key memory changes and improvements:

- Improved performance of memory allocation and de-allocation compared to previous RTX64 versions and Windows.
- Improved fragmentation to optimize memory usage within a process.
- Locks down page frames to reduce cache misses and guarantee cache coherence and TLB. Windows tradable memory will not impact the Subsystem or real-time processes.
- Provides more flexible configuration and control over how memory is used within the Subsystem and by user real-time processes.
- Allows for reserving memory on process startup to guarantee the memory you need will be available.
- Redesigned the Control panel to provide an overview of memory usage along with the ability to configure Subsystem, Network, and process MSpace settings.
- Added the ability to tools and the debugger to override default settings for initial reserved memory size and expanding of a process MSpace.
- Expanded the Real-Time API (RTAPI) to include calls to query and configure process MSpaces.
- Expanded the Managed and Native framework APIs to support easy configuration of local memory.
- Added a new RTX64 MSpaces command line utility, which displays local memory allocation spaces (MSpaces) of all RTSS processes (optionally including internal system processes and proxy processes).
- Added WinDbg extension commands for displaying local memory allocation space information.
- Added a new sample, Dynamic MSpace Configuration, which provides sample code, building as RTSS processes and Windows processes, respectively, that you can use to avoid memory request errors when memory allocation spaces (MSpaces) are configured to not expand automatically when exhausted (*Auto expand MSpace* is turned off in the Control Panel).

Network and Drivers

- Integrated the Network Abstraction Layer (NAL). The NAL is a network layer that abstracts the network hardware and driver functions from the upper-level protocol stacks and provides management interfaces for those upper layers to easily query for and use available network assets.
- Layered the RT-TCP/IP stack on top of the NAL to take advantage of the NAL's ability to better handle NIC resources.
- Re-architected the Virtual Network to provide a more robust connection and better performance.
- Improved error handling on startup of the real-time networking components. (8297)
- Expanded the Real-time Network (RTN) API, NIC Driver Real-time Network Device (RTND) API, and added new Real-time Network Abstraction Layer (RtNal) functions and structures.

Tools and Utilities

- Reworked the Control Panel to support integration of Network Abstraction Layer (NAL) functionality, layering of the RT-TCP/IP Stack on top of the NAL, and expanded memory allocation functionality.

Runtime Improvements

Network and Drivers

- RTSS applications can now call both a Network Abstraction Layer (NAL) API and an RT-TCP/IP Stack API at the same time. (7877)
- Increased the maximum number of sockets from 255 to 1024. (7879)
- Increased the device name length in the RT-TCP/IP Stack and drivers to 64 bytes. (8416)
- Added support for the Intel i219 LM4 Ethernet Connection NIC to the RtNalIPCH driver. See the *RTX64 Supported NICs* document for details. (7083)
- Improved error handling on startup of the Real-time network components. (7751)
- Added support for interrupt moderation to the RtNalIGB, RtNalIPCH, RtNalI10GB drivers. See the TechNote *Using Interrupt Moderation with Supported RTX64 Drivers* for instructions on using this feature. (5343)

Tools and Utilities

- Added a *Set maximum concurrency* setting to the Control Panel and framework that allows you to specify the number of threads that are allowed to run concurrently within the TCP/IP Stack. Historical information is made available to inform you of resource usage to help you optimize your system. (7952)
- Added settings to RtssRun and Task Manager that allow you to set an initial size (*/i*) and expand (*/e*) the process external MSpace for new and scheduled tasks. (8153)
- Added a */c* setting to RtssRun that checks compatibility of an RTSS and RTDLL binary with the target Runtime and lists any compatibility issues.
- RTX64 Analyzer output now contains information on RTX64 Visual Studio extension installers (VSIX). (8147)

SDK Improvements

Application Development

- Added a new setting to the RTX64 Application and RTDLL project templates in Visual Studio to optionally include support for the Network Abstraction Layer (NAL) component. (7941)
- Added a new setting to the RTX64 Application and RTDLL project templates in Visual Studio to optionally include RtVision support. Note that this requires the RtVision SDK to be installed. (8526)

Application Debugging

- Added new WinDbg extension commands for displaying local memory allocation space information:
    - !rtims / !rtems display summary and statistical information for the internal memory allocation space (MSpace) of a process.
    - !rtimspool / !rtemspool display local pool information for the process internal memory allocation space (MSpace) of a process.
    - !rtimspoolfg / !rtemspoolfg display fragmentation information within the local pool of an internal memory allocation space (MSpace) of a process.
    - !rtimspoolcache / !rtemspoolcache display chunk information within the pool cache of an internal memory allocation space (MSpace) of a process.
- Added support for WinDbg object string Mvector Interrupt, which specifies multiple interrupt vector attaching information. (5828)
- Added a new WinDbg extension command, !rtrdtinfo, which displays the system's RDT capability and CLOS configuration (if not ignored by RTX64).

- Improved the Real-Time Debugger to accept commit and expand size values for the local pool of a process's external allocation space (MSpace). (8158)

Real-Time APIs

- Added new Real-time APIs for querying and starting Subsystem components (8281):
  - RtQueryComponent returns status information for installed Subsystem components.
  - RtStartComponent starts a specified Subsystem component.
  - RTSSCOMPONENT specifies the type of networking component.
  - RT_COMPONENT_STATUS specifies the status of a Subsystem component.
- Added new macros for mapping specific functions (8033):
  - _tRtTtoi maps to functions RtAtoi and RtWtoi
  - _tRtPrintf maps to functions RtPrinf and RtWPrintf
- Added new RtNal structure RTNAL_INTERFACE, which specifies interface data for function RtndInitializeInterface.
- Added new Real-time function RtIsDefaultLocalMemory, which returns the default memory allocation configuration (local or Windows) for the Subsystem and RTSS applications.
- Added attribute values to Real-Time function RtUpdateProcThreadAttribute.
- Added new Real-Time APIs and changed existing APIs to support the new local memory architecture in RTX64:
  - Added RtGetProcessMSpace, which retrieves descriptors for the external and internal memory allocation spaces (MSpaces) for the specified RTSS process or Subsystem.
  - Added RtQueryProcessMSpace, which queries memory allocation space information in the specified process MSpace, including various summary statistics of its local pool and pool cache.
  - Added RtExpandMSpace, which expands the specified MSpace by the size specified.
  - Added RtShrinkMSpace, which shrinks the specified MSpace by the size specified.
  - Added structure MSPACE_INFO, which contains information for a memory allocation space (MSpace).
  - Changed RtAllocateLocalMemory to allocate memory from the external MSpace of the current process.
  - Changed RtAllocateLocalMemoryEx to allocate memory from the external MSpace of current process.
  - Updated event MF_EVENT_KIND_PROCESS_CREATE in structure MF_EVENT_KIND to include information on Local Memory MSpaces. (8154)
- Added support for members ReceiveLinkSpeed and TransmitLinkSpeed in the IP_ADAPTER_ADDRESSES structure:
  - ReceiveLinkSpeed is the current speed in bits per second of the receive link for the adapter.
  - TransmitLinkSpeed is the current speed in bits per second of the transmit link for the adapter.

- Added new Real-Time Network (RTN) functions:
  - RtnFrameAllocate is used by the filter driver, and a RTSS application with RT-TCP/IP support enabled, to allocate a block of memory to store an Ethernet frame.
  - RtnFrameFree frees memory allocated by RtnFrameAllocate.
  - RtnFrameTransmit is used by a filter driver or an application to transmit an Ethernet frame for the device whose source MAC address is in the frame.
  - RtnFrameTransmitInterface is used by a filter driver or an application to transmit an Ethernet frame for the device pointed by ndp. (3969)
  - RtnQueryStackHeapUsage retrieves heap usage information for the TCP/IP Stack. (5482)
  - RtnNotifyRecvQueue is used to notify the receiving application that data has been received.
  - RtnTransmitCompleteCallback must be called for a NAL frame transmitted by RtndTransmitEx.
- Added new NIC Driver Real-Time Network Device (RTND) functions:
  - RtndAttachToReceiveQueue attaches a NAL client application to the driver receive queue to start receiving packets.
  - RtndAttachToTransmitQueue (optional) attaches a NAL client application to the driver transmit queue to allow application transmit complete callbacks.
  - RtndDetachReceiveQueue detaches the driver receive queue from a NAL client application.
  - RtndDetachTransmitQueue (optional) detaches driver transmit queue from a NAL client application.
  - RtndReceiveWithCallback (optional) receives a single received packet from the driver. Packet data is available in an application callback.
  - RtndServiceTransmitQueue (optional) is called by the NAL to service transmitted packets in the context of a NAL client application.
  - RtndTransmitEx (optional) is a zero-copy function called to transmit multiple packets.
  - RTND_MEDIA_CONNECT_STATE is used in RtndRequest, RtNalRequest, and RtnRequest with RTND_OID_GEN_MEDIA_CONNECT_STATUS OID to get the status of the interface Ethernet connection.
  - RTND_MEDIA_DUPLEX_STATE is used in RtndRequest, RtNalRequest, and RtnRequest with RTND_OID_GEN_MEDIA_DUPLEX_STATE OID to get the duplex of the interface Ethernet connection
  - RTND_MEDIA_SPEED is used in RtndRequest, RtNalRequest, and RtnRequest with RTND_OID_GEN_LINK_SPEED OID to get the speed of the interface Ethernet connection.
- Added new Real-Time Network Abstraction Layer (RtNal) functions and structures:
  - RtNalEnumInterfaceInfo retrieves information on Network Abstraction Layer (NAL) interfaces. (4504)
  - RtNalGetInfo retrieves the current NAL settings.
  - RTNAL_INFO is a structure that specifies information to RtNalGetInfo about the current Network Abstraction Layer (NAL) settings.
  - RTNAL_INTERFACE is a structure that specifies interface data for functions RtndInitializeInterface and RtNalEnumInterfaceInfo.

- RTNAL_PHYSICAL_ADDRESS represents the physical address in NAL and NIC drivers.
- RTNAL_ETHERTYPE_FILTER is a parameter to RtndIoctl to set, get, or remove an Ethernet type filter with ioctls RTNAL_IOCTL_SET_RX_ETHERTYPE_FILTER, RTNAL_IOCTL_GET_RX_ETHERTYPE_FILTER, and RTNAL_IOCTL_CLEAR_RX_ETHERTYPE_FILTER.
- RTNAL_PTP_SET_TIMESTAMP_TYPE_ARGUMENTS is a parameter for RtndIoctl to set the PTP Receive packet type to be timestamped. It is used with ioctls RTNAL_IOCTL_SET_RX_MESSAGE_TYPE_TO_TIMESTAMP and RTNAL_IOCTL_SET_TX_MESSAGE_TYPE_TO_TIMESTAMP.

Native Framework APIs

- Added a new enumeration RTFW_COMPONENT_STATUS, which represents the status of an RTX64 component. This enumeration replaces the RTFW_SUBSYSTEM_STATUS enumeration from versions previous to RTX64 4.0.
- Added new Real-Time Native Framework functions and structures:
  - RtfwStartNAL starts the Network Abstraction Layer (NAL).
  - RtfwStopNAL stops the Network Abstraction Layer (NAL).
  - RtfwStartTCPIPStack starts the RT-TCP/IP Stack.
  - RtfwStopTCPIPStack stops the RT-TCP/IP Stack.
  - RtfwGetNALStatus returns the status of the Network Abstraction Layer (NAL).
  - RtfwGetTCPIPStatus returns the status of the RT-TCP/IP Stack.
  - RtfwGetNALConfiguration returns the configuration of the Network Abstraction Layer (NAL).
  - RtfwSetNALConfiguration configures the Network Abstraction Layer (NAL).
  - RTFW_NAL_CONFIGURATION represents the configuration of the Network Abstraction Layer (NAL). This structure replaces the RTFW_NETWORK_CONFIGURATION structure from versions previous to RTX64 4.0.
  - RtfwGetTCPIPConfiguration returns the configuration of the RT-TCP/IP Stack.
  - RtfwSetTCPIPConfiguration configures the RT-TCP/IP Stack.
  - RTFW_TCPIP_CONFIGURATION represents the configuration of the RT-TCP/IP Stack. This structure replaces the RTFW_NETWORK_CONFIGURATION structure from versions previous to RTX64 4.0.
  - RtfwGetNetworkVerbosity returns whether verbosity is enabled or disabled for the network (NAL and TCP/IP Stack).
  - RtfwSetNetworkVerbosity enables or disables verbosity across the network (NAL and TCP/IP Stack).
- Added new Real-Time Native Framework APIs for configuring Local Memory (8303, 8493):
  - RtfwGetLocalMemoryConfiguration returns the Local Memory MSpaces configuration for the RTX64 Subsystem.
  - RtfwSetLocalMemoryConfiguration sets the Local Memory MSpaces configuration for the RTX64 Subsystem.
  - RtfwGetNALLocalMemoryConfiguration returns the local memory configuration of the Network Abstraction Layer (NAL).
  - RtfwSetNALLocalMemoryConfiguration specifies the local memory configuration of the Network Abstraction Layer (NAL).

    - RtfwGetTCPIPLocalMemoryConfiguration returns the local memory configuration of the RT-TCP/IP Stack.
    - RtfwSetTCPIPLocalMemoryConfiguration specifies the local memory configuration of the RT-TCP/IP Stack.
    - RTFW_LOCAL_MEMORY_CONFIGURATION represents the configuration of the RTX64 Subsystem and its use of Local Memory MSpaces.
    - RTFW_NETWORK_LOCAL_MEMORY_CONFIGURATION represents the configuration of the real-time network and its use of Local Memory MSpaces.
- Added new members to Real-Time Native Framework structure RTFW_SCHEDULED_PROCESS (8307):
    - LocalMemoryCommitSize specifies the size, in kilobytes, by which to allocate Non-Paged memory from Windows into the Local Pool of the MSpace at process startup.
    - LocalMemoryExpandSize specifies the size, in kilobytes, by which to expand an RTSS processes' internal/external MSpaces.

Managed Framework APIs

- Added new properties to Managed Framework class IntervalZero.RTX64.Config.ScheduledProcess (8307):
    - MSpaceInitialSize gets/sets the size, in kilobytes, by which to allocate Non-Paged memory from Windows into the Local Pool of the MSpace at process startup.
    - MSpaceExpandSize gets/sets the size, in kilobytes, by which to expand an RTSS processes' internal/external MSpaces.
- Added new APIs to Managed Framework class IntervalZero.RTX64.RTAPI.RtssEnvironment (8320):
    - Method GetProcessMSpace retrieves a descriptor of the external and internal memory allocation space (MSpace) for the specified RTSS process or system process.
    - Method QueryProcessMSpace queries memory allocation space information for the specified process MSpace, including various summary statistics of its local pool and pool cache.
    - Structure MSPACE_INFO contains information for a single memory allocation space (MSpace).
- Added new local memory configuration properties to Managed Framework class IntervalZero.RTX64.Config.Subsystem (8303, 8493):
    - EnableLocalMemory accesses the configuration parameter that specifies whether the RTX64 Subsystem uses Local memory (MSpaces) or Windows memory. If true, the real-time Subsystem memory allocation uses Local memory MSpaces. If false, the Subsystem allocates memory by requesting it from Windows.
    - NALExtMSpacePoolMinimumSize accesses the configuration parameter that specifies the minimum size, in kilobytes, of the local pool within the Network Abstraction Layer (NAL) external MSpace.
    - NALMSpacesPoolExpandable accesses the configuration parameter that specifies whether the local pool within the Network Abstraction Layer (NAL) MSpaces is expandable. If true, the local pool within the NAL MSpace is expandable by the amount specified by NALMSpacesPoolExpandSize when depleted.

- NALMSpacesPoolExpandSize accesses the configuration parameter that specifies the size, in kilobytes, by which to expand the local pool within the Network Abstraction Layer (NAL) MSpaces when NALMSpacesPoolExpandable is set to true.
- ProcessExtMSpacePoolMinimumSize accesses the configuration parameter that specifies the minimum size, in kilobytes, of the local pool within the RTSS process external MSpace.
- ProcessIntMSpacePoolMinimumSize accesses the configuration parameter that specifies the minimum size, in kilobytes, of the local pool within the RTSS process internal MSpace.
- ProcessMSpacesPoolExpandable accesses the configuration parameter that specifies whether the local pool within the RTSS process MSpace is expandable. If true, the local pool within the RTSS process MSpace is expandable by the amount specified by ProcessMSpacesPoolExpandSize when depleted.
- ProcessMSpacesPoolExpandSize accesses the configuration parameter that specifies the size, in kilobytes, by which to expand the local pool within the RTSS process MSpace when ProcessMSpacesPoolExpandable is set to true.
- ProcessMSpacesPoolShrinkable accesses the configuration parameter that specifies whether the local pool within the RTSS process MSpace is shrinkable.
- SystemExtMSpacePoolMinimumSize accesses the configuration parameter that specifies the minimum size, in kilobytes, of the local pool within the system process external MSpace.
- SystemIntMSpacePoolMinimumSize accesses the configuration parameter that specifies the minimum size, in kilobytes, of the local pool within the system process internal MSpace.
- SystemExtMSpacePoolCommit accesses the configuration parameter that specifies whether the local pool within the system process external MSpace is committed at startup.
- SystemMSpacesPoolExpandable accesses the configuration parameter that specifies whether the local pool within the system process MSpace is expandable by the amount specified by SystemMSpacePoolExpandSize when depleted.
- SystemMSpacesPoolExpandSize accesses the configuration parameter that specifies the size, in kilobytes, by which to expand the local pool within the system process MSpace when SystemMSpacePoolExpandable is set to true.
- SystemMSpacesPoolShrinkable accesses the configuration parameter that specifies whether the local pool within the system process MSpace is shrinkable.
- TCPIPExtMSpacePoolMinimumSize accesses the configuration parameter that specifies the minimum size, in kilobytes, of the local pool within the RT-TCP-IP Stack external MSpace.
- TCPIPMSpacesPoolExpandable accesses the configuration parameter that specifies whether the local pool within the RT-TCP/IP Stack MSpace is expandable. If true, the local pool within the RT-TCP/IP Stack MSpace is expandable by the amount specified by TCPIPMSpacesPoolExpandSize when depleted.

    - TCPIPMSpacesPoolExpandSize accesses the configuration parameter that specifies the size, in kilobytes, by which to expand the local pool within the RT-TCP/IP Stack MSpace when TCPIPMSpacesPoolExpandable is set to true.
    - ZeroMemoryAtAllocation accesses the configuration parameter that specifies whether the memory allocated by malloc or similar C-Runtime functions is initialized to zero (according to the C99 specification).
- Added new method SetHALTimerPeriod to Managed Framework class IntervalZero.RTX64.Config.Subsystem, which sets the HAL timer period (in microseconds) for the RTX64 Subsystem.
- Added new method ResetToStart to Managed Framework class IntervalZero.RTX64.Monitor, which resets the event reader so the next call to EventReader.ReadEvents returns event(s) from the start of the monitoring session. (8725)
- Added new methods to Managed Framework class IntervalZero.RTX64.Control.Subsystem:
    - GetNALStatus queries the status of the Network Abstraction Layer (NAL).
    - GetTCPIPStatus queries the status of the RT-TCP/IP Stack.
- Added a new class to Managed Framework namespace IntervalZero.RTX64.Control, NetworkInterfaceStatus, which contains these new properties:
    - Name indicates the name of the network interface.
    - NALInterfaceStatus indicates the status of the NAL interface.
    - TCPIPInterfaceStatus indicates the status of the TCP/IP interface.
    - ErrorCode holds the native Windows or RTX64 custom error code, if either NALInterfaceStatus or TCPIPInterfaceStatus is ComponentStatus.Error.

RTX64 3.7

General

- Added support for Windows 10 19H1 Feature Update version 1903. (7360)
- Improved RTSS processor startup logic to better support newer processors on embedded hardware. (8082)

Tools and Utilities

- RTX64 Analyzer output now contains information on RTX64 Visual Studio extension installers (VSIX). (8147)

Application Development

- Added development and debugging support for Visual Studio 2019. (8080)
- Added support for the ANSI string convention to the RTX64 RTSS/RTDLL project templates. (6436)

RTX64 3.6

General

- Added support for Windows 10 October Feature Update version 1809 and Long Term Servicing Channel (LTSC) version 2019. (7507)

Configuration

- Added a Subsystem validation feature that validates the RTSS boot configuration and warns when changes to the RTSS boot configuration will cause conflicts with existing ideal processor and affinity mask assignments. (7250)

Tools and Utilities

- Upgraded the Percepio Tracealyzer diagnostic tool to version 4, which includes:
    - Revamped UI with added docking support for windows/views.
    - A welcome page with a list of recent sessions.
    - Improved Trace View to support common functionality in both vertical and horizontal views.
    - Support for intervals and state machines.
    - Query-based Finder window.
    - The ability for Tracealyzer to display application pathnames for Windows proxy processes. (5414)
- Made improvements to monitoring and the Monitor utility:
    - Added support for monitor session files (.monx). (5465)
    - Changed the open session dialog to open a file (.monx, .mev) rather than a folder. (7492)
    - Added functionality to the Monitor utility and Control Panel that allows you to check all or clear all events. (6275, 7575)

- Added functionality to the Monitor utility that allows you to pause a monitoring session. The Control Panel displays a Monitoring paused status when the current session is in a paused state. (7479, 7480)
- Added new monitoring events:
  - Watchdog Reset – represents an instance where a watchdog timer reset occurs on certain cores. (6967)
  - Heap Re-Allocate – represents a successful call to real-time API HeapReAlloc. (6552)
  - Thread Open – represents the instance a real-time thread is opened. (5914)
  - CLOS Set – represents a change in Class of Service (CLOS) of a thread. (6807)
- Added support for enabling/disabling monitor event generation by product component. Currently, this is only controls event generation for the RT-TCP/IP Stack. (2798)
- Added functionality that allows you to toggle inclusion of network-related events in a monitoring session. This can either be done on a persistent level, through the Control Panel, or on a transient (per session) level, through the Monitor utility. (7567)

Application Development

- Added support for Visual Studio 2017 version 15.9. (7652)

SDK

- Real-Time
  - Increased the maximum number of objects supported by RtWaitForMultipleObjects from 16 to 64. (7371)
  - Added new Real-Time functions and enumerations to support enabling/disabling of monitor event generation by product component (2798):
    - RtMonitorGetEnabledComponents – retrieves the transient configuration of RTX64 product components that currently have monitoring event generation enabled.
    - RtMonitorEnableComponents – transiently enables the generation of monitor events by one or more RTX64 product components.
    - RTX64_MONITOR_COMPONENT – an enumeration that represents RTX64 product components that can have their monitoring event generation enabled/disabled transiently.
  - Added new Real-Time functions and structures to support setting event types and custom event types as triggers to start monitoring:
    - RtMonitorSetCustomTriggers – transiently sets the custom event types that will trigger the start of a monitoring session.
    - RtMonitorSetTriggers – transiently sets the event types that will trigger the start of a monitoring session.
    - RT_MONITOR_CUSTOM_TRIGGER_CONTROL – Transiently associates trigger values to custom monitoring event types.

        - RT_MONITOR_TRIGGER_CONTROL – Transiently associates trigger values to monitoring event types.
    - Added new Real-Time function, RtMonitorChangeState, which allows you to transition between monitoring states.
    - Added a new Real-Time function RtMonitorSetEvents , which enables or disables a collection of events during a monitoring session.
    - Added constants to Real-Time enumeration RTX64_MONITOR_CONTROL_OP:
        - MONITOR_CONTROL_ENABLE_COMPONENTS – transiently enables generation of monitoring events by one or more RTX64 product components.
        - MONITOR_CONTROL_GET_ENABLED_COMPONENTS – retrieves the transient configuration of RTX64 product components that currently have monitoring event generation enabled.
        - MONITOR_CONTROL_PAUSE – pauses collection of monitoring events during the current session without having to create a new session file.
        - MONITOR_CONTROL_RESUME – resumes the monitoring session from a Paused state.
- Managed Code Framework
    - Added new class MonitorEventThreadOpen under IntervalZero.RTX64.Monitor, which represents the instance a real-time thread is opened. This occurs when a real-time process calls OpenThread. (5914)
    - Added new class MonitorEventHeapReAlloc under IntervalZero.RTX64.Monitor, which represents a successful call to HeapReAlloc. (6552)
    - Added new class MonitorEventWatchdogReset under IntervalZero.RTX64.Monitor, which represents an instance where a watchdog timer reset occurs on certain cores. (6967)
    - Added new class MonitorEventCLOSSet under IntervalZero.RTX64.Monitor, which allows you to set a new Class Of Service (CLOS) for a thread. (6807)
    - Added new method GetEventReader to the EventReader class under IntervalZero.RTX64.Monitor, which opens a monitor session file (.monx) or a monitor event file (.mev), which opens the Event Reader session. (5465)
    - Added new methods and enumerations to class IntervalZero.RTX64.Monitor.Subsystem to support enabling/disabling of monitor event generation by product component (2798):
        - GetEnabledComponents – this method returns the transient or persistent configuration specifying which product components currently have monitor event generation enabled
        - EnableComponents – this method transiently or persistently enables monitor event generation for one or more product components.
        - MonitorComponent – this enumeration represents the product components that can have monitor event generation enabled/disabled transiently or persistently.
    - Added new method SetWindowsAndRtssProcessorsRequested to class IntervalZero.RTX64.Config.Subsystem, which atomically sets the requested number of Windows and RTSS processors. (7573)

RTX64 3.5

General

- Added support for Secure Boot on Windows 10. (7031)
- Added support for C++ magic static local variables and implicit Thread-Local Storage (TLS), which supports initialization and finalization of variables declared with storage class *thread_local* and *__declspec(thread)*. (5885, 6089, 7033)

Subsystem

- Added support for performance optimization with Intel® Resource Director Technology (RDT) resource allocation capabilities, including Cache Allocation Technology (CAT) and Memory Bandwidth Allocation (MBA) configurations. See SDK below for a list of Real-time functions and configuration functions that have been added to support this functionality. (6460, 6043)
- Improved Watchdog Timer support. Modified internal calculation so dedicated RTSS cores correctly reset on a context switch, not the processor idle thread. See SDK below for information on the newly provided API to manually reset a watchdog timer for given cores along with configuring the feature within the Subsystem. (4430)
- Added support for overriding Windows Energy/Performance Bias, along with a range for configuring performance vs. energy savings in RTX64. (6449)
- Added support for GS segment registers, which enable use of:
  - Windows SDKs newer than version 8.1
  - C++ magic static local variables and implicit Thread-Local Storage (TLS), which supports initialization and finalization of variables declared with storage class *thread_local* and *__declspec(thread)*.

  (5885, 6089, 7033, 6111)

RT-TCP/IP Stack and Drivers

- Added support for the Matrox Concord GE (customized Intel® 82574) Ethernet Controller (0x10D3) to the Intel® RtE1000 driver. See the RTX64 Supported NICs document for more information: https://www.intervalzero.com/technical-support/guides-and-minitutorials/ (6983)

Tools and Utilities

- Added the following enhancements to the Task Manager:
  - Task Manager now provides CPU utilization information at a per-process, -thread, and -processor level. (6340)
  - Task Manager now displays a status (Running, Under Debug, Suspended, Frozen) for each listed process. (2816)
  - Task Manager now displays the Ideal Processor for listed RTSS processes. (6263)
  - You can now type the name of an .rtss file or .exe file in the Start New Task dialog to start a process, in addition to browsing to the file. (2817)
  - The current sort order is now saved when the tool is closed and retained when it is next opened. (6729)
- Added the following enhancements to the RTX64 Analyzer:
  - Analyzer output now includes information on the Windows updates installed on the system (7034).
  - Analyzer output now includes information on the status of performance optimization with Intel® Resource Director Technology. (6763)

Application Development

- Added support for secure C Runtime functions *assert* and *abort* under Debug mode that cause assertions when invalid parameters are passed. Visual Studio templates have been modified to include a new debug StartupCRTd library under the linker options in RTSSDebug configurations to handle the assertions. Assertion checks that fail through secure functions will throw an exception and display a message through the RTX64 Server console. (2688, 7019, 7020)

SDK

- Real-Time
  - Added new Real-Time functions for configuring performance optimization with Intel® Resource Director Technology (RDT) resource allocation:
    - RtGetRDTCapability – returns the system Resource Director Technology (RDT) allocation capability.
    - RtGetThreadCLOS – retrieves the Class of Services (CLOS) value of a specified thread.
    - RtSetThreadCLOS – overwrites a thread's implicit or default Class of Services (CLOS), which is based on its priority.
  - Added a new Real-Time function, RtGetProcessIdealProcessor, which retrieves the ideal processor for a given RTSS process handle. (2079)
  - Added a new Real-Time function, RtResetWatchdog, which resets the watchdog timer counter on given RTSS cores. (4424)
  - Added a new Real-Time function, RtGetProcessTimes, which retrieves timing information for the specified RTSS process. (2815)

- Added a new Real-Time function, RtGetSystemTimes, which retrieves an array of system timing information. (2815)

- Managed Code Framework
  - Added a new Managed Framework property, WindowsEnergyPerformanceBias, which configures the Windows Processor Energy/Performance Bias functionality. (7066, 7069)
  - Added new Managed Framework properties to support configuring performance optimization with Intel® Resource Director Technology (RDT) resource allocation:
    - CATMode – retrieves the current Intel® Cache Allocation Technology (CAT) mode for performance differentiation among RTSS threads.
    - MBAMode – retrieves the current Intel® Memory Bandwidth Technology (MBA) mode for performance differentiation among RTSS threads.
    - RDT – retrieves the current state of Intel® Resource Director Technology (RDT) performance optimization.
  - Added two new Managed Framework enumerations that represent the status of performance optimization with Intel® Resource Director Technology (RDT) resource allocation:
    - RDTState – this enumeration represents the state of Intel® Resource Director Technology (RDT) performance optimization.
    - RDTFeatureMode – this enumeration represents the mode of the Intel® Resource Director Technology (RDT) features.
- Native Framework
  - Added two new Native Framework functions that set/get the configuration for the Windows Energy/Performance Bias feature. (7066, 7069)
    - RtfwGetWindowsEnergyPerformanceBias – reads the current configuration for the WindowsEnergyPerformanceBias feature.
    - RtfwSetWindowsEnergyPerformanceBias – sets the configuration for the WindowsEnergyPerformanceBias feature.
  - Added two new Native Framework enumerations that represent the status of performance optimization with Intel® Resource Director Technology (RDT):
    - RTFW_RDT_STATE – this enumeration represents the state of Intel® Resource Director Technology (RDT) performance optimization.
    - RTFW_RDT_FEATURE_MODE – this enumeration represents the mode of the Intel® Resource Director Technology (RDT) features.
  - Expanded the Native Framework RTFW_SUBSYSTEM_CONFIGURATION structure to include RDT resource features. Calls to RtfwGetSubsystemConfiguration and RtfwSetSubsystemConfiguration can now be used for feature configuration.
  - Added new RTX64 Native Framework Library functions for managing scheduled tasks (6680):
    - RtfwCreateScheduledProcess – creates a new scheduled process to start with the Real-time Subsystem.
    - RtfwDeleteAllScheduledProcesses – deletes all scheduled processes.
    - RtfwDeleteScheduledProcessByID – deletes the scheduled process specified by an identifier.

- RtfwGetCurrentScheduledProcessCount – retrieves the total number of scheduled processes.
- RtfwGetPIDByScheduledID – retrieves the last known PID of the scheduled process that is currently running.
- RtfwGetScheduledProcess – retrieves information about a previously scheduled process.
- RtfwModifyScheduledProcess – modifies an existing scheduled process.
- RTFW_SCHEDULED_PROCESS – a structure that holds the data fields for a single scheduled process.

Samples

- Added an RDTPerformance Sample, which provides an example of how to use RTX64-supported Intel® Resource Director Technology (RDT) to optimize the performance of particular RTSS threads with high performance requirements. (6828)
- Added a sample that demonstrates how to use FastSemaphores. (4243)

RTX64 3.4

General

- Added support for Windows 10 April 2018 Update Version 1803.
- Added support for the RTX64 Network Abstraction Layer (NAL).
- Product Help for RTX64 is now available in HTML5 format, replacing the CHM format previously installed. This new format provides a more contemporary look-and-feel, as well as improved search functionality.

Activation and Configuration

- Redesigned and added new functionality to the Activation and Configuration utility. You can now:
  - activate and manage licenses for RTX64 components and set the RTSS boot configuration from a single screen. (5848)
  - enable and disable licenses through the Activation and Configuration utility. (6258)
  - delete licenses from a license file on a hard drive and dongle. (5636)

Subsystem

- Added support for Intel® Advanced Vector Extensions 512 (AVX-512) instructions.

Tools and Utilities

- Added a System Tray application for RTX64 that displays the current state of the Subsystem and provides links to various RTX64 tools from a right-click menu.
- Improved Task Manager to allow for tasks to be scheduled to start automatically with the Subsystem. (1011)

SDK

- Added the RTX64 Native Framework (RtfwAPI.dll), a native DLL and import library that replicates the functionality of the Managed Code Framework.
- Added support for Windows applications to Real-time function RtMonitorControl. (5850)
- Added a new Real-time function, RtGetProcessorInfo, that retrieves the number values for the lowest and highest RTSS processors. (3178)

Samples

- Added a new sample, MulticoreResponseTimeMeasurements, which runs a System Response Timer on different cores simultaneously. (4427)
- Added a new sample, Native Framework Client, which demonstrates how to use the RTX64 Native Framework library in a Visual Studio C/C++ project.
- Added a new sample, RawIpSocket, which runs a raw IP echo client and/or raw IP echo server, depending on the specified command line options. (2687)
- Improved the TCPIP sample to include TCPIPClientNblock, which demonstrates how to use non-blocking connect. (4938)

RTX64 3.3

General

- Added support for Multiple Vector Interrupts in MSI-X. (1575)

Licensing

- Added the ability to activate RTX64 product components using a fingerprint file even when the machine is connected to the Internet. (4441)

SDK

- Added support for Windows applications to Real-time function RtMonitorControl. (5850)
- Expanded existing and added new Real-time functions to support Multiple Vector Interrupts (1575):
    - Added a new AttachVersion type, ATTACH_MESSAGE_BASED_MULTI_VECTOR, to the structure ATTACH_INTERRUPT_PARAMETERS used by RtAttachInterrupt, which allows the user to attach to message-based multiple vector interrupts.
    - RtGetPciMsixFreeMessages returns a PCI device's MSI-X free messages (the MSI-X table entries available for attaching messages).
    - RtQueryProcessorVectorFreeCount queries the number of RTSS processor vectors (Interrupt Descriptor Table vectors) currently available for attaching interrupts.
- Added a new Real-time function, RtOpenThread, which opens an existing thread object and returns a handle to that thread object. (4667)
- Added new Real-time functions for setting and retrieving APIC counts per RTX64 HAL timer period (5702):
    - RtGetHalTimerPeriodCounts
    - RtSetHalTimerPeriodCounts
- Added a new Real-time function, RtGetThreadStack, which retrieves the thread stack parameters for an existing thread object. (1639)

Samples

- Added the new sample MultiVectorI350, which provides an example of using Message Signaled Interrupts (MSI-X) with multiple vector attachment on Intel Ethernet Controller I350 with Dual or Quad ports. (1575)

RTX64 3.2

General

- Added support for the Intel® Apollo Lake platform. (5493)
- Added support for Windows 10 Creators Update Version 1703.

RT-TCP/IP Stack and Drivers

- Added support for multiple (up to 12) IPv4 Addresses and Netmasks for a single physical interface. You can now configure multiple IPv4 Address/Netmask pairings in the RTX64 Control Panel on the *Manage Interfaces and Filters* page. (5471)
- Incorporated the RTX64 Intel® PCH Network Interface (RTIPCH) driver, which provides support for the Intel i219 Network Interface Cards (NICs), as well as updated support for several Intel ICH8, ICH9, ICH10, and PCH controllers. This driver was previously released as a BETA driver on the Support site. See the *RTX64 Supported NICs* document for a list of supported devices: https://www.intervalzero.com/technical-support/guides-and-minitutorials/

Tools and Utilities

- Added new options to the Configure RT-TCP/IP Stack Behavior page in the RTX64 Control Panel:
  - Set the time-out on IP fragmentation
  - Set the maximum number of ARP entries allowed for the RT-TCP/IP Stack

  Previously, these values could only be changed in the Windows registry.

Samples

- Added new options to the RtTcpipClient and RtTcpipServer program samples to support multiple IPv4 addresses. (5600)
  - RtTcpipClient: *a=<ip> switch* – Specify the IP to bind the client socket to.
  - RtTcpipServer: */m* – Run multiple instances of the server.

RTX64 3.1

Subsystem

- Improved Subsystem performance by a range of 16%-33%, depending on system cache architecture. (987)

Tools and Utilities

- Improved monitoring event capture to gather information about processes running before monitoring is started. (5043)

- Implemented these improvements in the RTX64 Analyzer:
  - Analyzer output now contains a listing of the installed versions of .NET (5284)
  - Analyzer output now contains a listing of the RTX64 DLLs in the Global Assembly Cache (GAC). (5284)
  - Analyzer output now contains the contents of internal Registry keys. (5285)

Application Development and Debugging

- Added the feature Attach to RTSS Process, which allows you to attach the Visual Studio 2015 debugger to any running RTSS process on the local machine.
- Added support for Intel Compiler 17.0.1 (as shipped with Intel Parallel Studio XE 2017 Update 1). (5317)
- Added a new debugging property in Visual Studio that allows you to allocate memory from the Windows memory pool, which uses non-deterministically allocated memory. Note that this only applies when the default memory allocation behavior is set to use Local memory. (4257)
- Added BETA support for Visual Studio 2017 (RC2 was tested).

SDK

- Added new Real-Time Network functions:
  - RtnAttachProcessExitHandler registers an application's networking exit handler to allow an RTSS application to perform custom socket code cleanup when an application exits.
  - RtnReleaseProcessExitHandler removes an application's networking exit handler registered by the function RtnAttachProcessExitHandler. (5355)
- Added support for these Interlocked functions in real-time applications:
  - InterlockedAdd
  - InterlockedAdd64
  - InterlockedAnd
  - InterlockedAnd8
  - InterlockedAnd16
  - InterlockedAnd64
  - InterlockedCompare64Exchange128
  - InterlockedCompareExchange16
  - InterlockedCompareExchange64
  - InterlockedCompareExchange128

    - InterlockedCompareExchangePointer
    - InterlockedDecrement16
    - InterlockedDecrement64
    - InterlockedExchange8
    - InterlockedExchange16
    - InterlockedExchange64
    - InterlockedExchangePointer
    - InterlockedExchangeSubtract
    - InterlockedExchangeAdd64
    - InterlockedIncrement16
    - InterlockedIncrement64
    - InterlockedOr
    - InterlockedOr8
    - InterlockedOr16
    - InterlockedOr64
    - InterlockedXor
    - InterlockedXor8
    - InterlockedXor16
    - InterlockedXor64
- Added a new Real-Time API RtIsDebuggerPresent, which determines whether a local real-time process is attached to the IntervalZero Real-Time Debugger. (5417)
- Added support for these C Runtime functions:
    - errno
    - fwrite
    - fflush
    - ferror
    - feof
    - clearerr
- See the *Matrix of C Library Supported Functions* in the RTX64 Help for the full list. (5280)

RTX64 3.0

General

- Added support for Windows 10 Anniversary Update Version 1607.
- Added support for Windows 10 Version 1511 (Feb 2016). (4598)
- Added user notification pop-ups when a IntervalZero-dongle is plugged in and is available for use (4599)

RT-TCP/IP Stack and Drivers

- Added support for the Intel x540 10Gb/s network adapter (0x1528) through the RTX64 RtI10GB driver. This driver also includes untested support for several other Intel network adapters in the same family. See the RTX64 Supported NICs document, available at https://www.intervalzero.com/technical-support/guides-and-minitutorials/, for a list of supported devices. (4567)
- Incorporated the RtBCM driver, which had previously been released as a standalone. See the RTX64 Supported NICs document, available at https://www.intervalzero.com/technical-support/guides-and-minitutorials/, for a list of supported devices. (4566)
- Resolved performance issues that occurred when socket applications were run on gigabit connections and on at least two different cores. (4663)
- Added support for the Intel® I210 Flash-less Copper-only Ethernet Controller (device ID 0x157B) to the RtIGB driver. (4531)
- Added support for the Intel® PRO/1000 PF Dual Port Server Adapter (device ID 0x105F) to the RtE1000 driver. (4880)

Tools and Utilities

- Added Tracealyzer for RTX64, a diagnostics tool from Percepio for viewing monitoring session data, to the SDK. (4692)
- New Monitoring features:
    - Improved monitor event MonitorEventSemaphoreRelease to include the handle of the semaphore. (4314)
    - Improved monitor events MonitorEventWFSOReturned and MonitorEventWFMOReturned to return a pointer to the objects they acted upon. (4315)
    - Expanded monitor events to include handles for Release events. (4312)
    - Improved Critical Section monitor events to include the mutex handle of the critical section object. (4701)
    - Added these new monitor events: (4904, 4432)
        - File Object Create – Generated when a file has been opened by the subsystem.

        - File Object Destroy – Generated when a file has been destroyed by the subsystem.
        - SRI to Windows Event – Generated when an SRI over to Windows has been initiated.
        - SRI to Windows Return Event – Generated when an SRI over to Windows has returned.
        - Thread Terminate – Generated when a thread terminates.
- Improved the Latency View tool to allow each graph view to be scaled independently and to log the duration of the test run. (4414, 1668, 3548)
- Added the following information to RTX64 Analyzer output:
    - System boot configurations. (4798)

      NOTE: In order to display boot configuration data when running RTX64 Analyzer from the Start Menu shortcut, you must right-click and choose Run as administrator.

    - Monitor configuration information (4792)
    - VSIX Extensions installed on the machine (4784)
    - On Windows 8.1 systems: whether the supported Windows 8.1 version (Windows 8.1 with Update) is installed. (4793)

SDK

- Integrated standalone Visual Studio 2015 with Update 2 build and debug support, including (4844, 4530):
    - Visual Studio templates for creating a RTX64 Application and RTDLL.
    - Support for statically linked Debug and Release versions of the Microsoft Visual Studio C Runtime.
    - Added Snippets for some key RTAPI function calls.
    - Added debugging support through launch.
    - Added support for Start Without Debugging within the Visual Studio debugger.
    - Added support for launching a RTSS process on a remote target system for debugging.
- Added support for Intel Compiler 16.0. (4396, 4406)
- Added custom error codes that can be returned by Real-Time APIs. (4601, 4459)
- Added a new Real-Time function RtGetModuleFileNameEx, which retrieves the fully-qualified path for the file that contains the specified module. (4470)
- Added new RTK API functions that verify whether a specified RTX64 Runtime or RT-TCP/IP Stack version is installed and licensed (4660):
    - RtkIsRuntimeLicensed – Verifies that a specified RTX64 Runtime is installed and has a valid license.
    - RtkIsTcpStackLicensed – Verifies that a specified RTX64 RT-TCP/IP Stack component is installed and has a valid license.
- Added new API functions that return whether the provided RTSS application binary can be run, which means it has been built to run with the provided license feature and that there is a valid license for the feature on the system. (4643)

    - RtIsAppRunnable
    - RtkIsAppRunnable
- Added new API functions that return the version of the installed RTX64 Runtime (3151):
    - RtGetRuntimeVersionEx
    - RtkGetRuntimeVersionEx
    - RT_VERSION_INFO
- Added support for the Windows-supported API function GetModuleHandleEx, which retrieves a module handle for the specified module in a specified process. (3176)
- Added support for these C-Runtime functions: (4541)
    - _beginthread, _beginthreadex
    - _endthread, _endthreadex
- Added new Real-Time API functions for setting and retrieving the flush TLB tick mod of a RTSS process. By default, a processor's TLB cache is flushed when the processor is idle (4690):
    - RtSetFlushTLBTickMod
    - RtGetFlushTLBTickMod
- Added new Real-Time API functions for setting and retrieving the time quantum value for a specified thread, in microseconds (4714):
    - RtSetThreadTimeQuantumEx
    - RtGetThreadTimeQuantumEx
- Added support for Winsock function WSAEnumNetworkEvents, which discovers occurrences of network events for the indicated socket. Additionally, support was added for structure WSANETWORKEVENTS, used to store a socket's internal information about network events for WSAEnumNetworkEvents. (4703)
- Improved performance in the Real-Time Network Device function RtndFrameTransmit. (4688)
- Added a new Real-Time API function, RtEnumProcessEx, which retrieves the process identifier for each RTSS process object using data specified by new structure RTPROCESS_INFORMATION. (4775)

Samples

- Added a Start Menu entry that points directly to the location to which Samples are installed by default: %public%\Documents\IntervalZero\RTX64 SDK\\Samples. You can also access RTX64 program samples directly from the RTX64 SDK Welcome Screen. (3579, 4564)

- Updated the RTKIPC sample, which now requires Visual Studio 2015 and WDK 10 to build. The resulting binary will run on all Windows versions supported by RTX64 (7, 8.1, and 10). As part of this update, Visual Studio 2012 and Visual Studio 2013 versions of the sample are no longer provided. (4877, 4885)
- Added an Advanced Installer sample to the Merge Modules installation showing how to use the RTX64 Runtime merge modules. (4966, 4705)

RTX64 2014 with Service Pack 2

General

- Added support for Windows 8.1 with Update (3220, 3221).

Activation and Configuration

- Added support for smaller form factor dongles that can hold license files. (3394)
- Added functionality to provide dongle information through command line utilities and API calls. (3880)

Subsystem

- You can now configure a Search Path to allow RTX64 to load an RTSS application or RTDLL by base filename only, assuming one of the directories in the search path contains the file you want to load. (3895, 3709, 2027)

Tools and Utilities

- Added a new RTX64 CPU Usage utility, which displays CPU usage information for all RTSS cores on the system. (3782)
- Added a new RTX64 Objects utility, which displays information about RTSS processes and their associated objects, such as events, semaphores, and loaded RTDLLs. (3454)

Monitoring

- Provided the ability to associate a monitor event with a trigger to start monitoring (2971)
- Improved the Timer Set monitoring event to include fields for Expiration Period and Interval Period. (3302)

- Redesigned the Monitor utility and added support for the optional association of monitor events with triggers. (3376)
- Added new memory free failure events to the Monitor utility (3410):
    - Local Memory Free Fail – Represents failed freeing of local memory.
    - TLS Free Fail – Represents a failed call to TlsFree by a user application.
    - Heap Free Fail – Represents failed freeing of memory via HeapFree().
    - Contiguous Memory Free Fail – Represents failed freeing of contiguous memory
    - Windows Memory Free Fail – Represents failed freeing of memory to Windows.

RT-TCP/IP Stack and Drivers

- Added support for the Intel Gigabit ET2 Quad-Port driver. (3323)
- Added support for the Intel i217 and i218 Network Interface Cards (NICs). (3711, 3228, 3100)

Debugging

- Added new debugging properties to application Property Pages in Visual Studio (3848). You can:
    - Choose to override default subsystem behavior to allocate memory from the RTX64 local memory pool, which uses deterministically allocated memory.
    - Set the ideal processor on which the main thread of the debugged process will run.
    - Set the affinity mask that specifies the processor(s) on which the debugged process will run.

SDK

- Added new API functions that list the serial number for each dongle connected to the machine. (2676)
    - RtGetDongles
    - GetDonglesInfo (available from the Managed Code Framework)
- Added two new API functions that retrieve information about licenses installed on the system. (3034)
    - RtGetLicenses
    - RtkGetLicenses
- Added new API functions that verify whether a specified RTX64 Runtime or RT-TCP/IP Stack version is installed and licensed:
    - RtIsRuntimeLicensed – Verifies that a specified RTX64 Runtime is installed and has a valid license.
    - RtIsTcpStackLicensed – Verifies that a specified RTX64 RT-TCP/IP Stack component is installed and has a valid license.

- Added support for an IP Helper function and structures (3536):
  - GetAdaptersAddresses, which retrieves the addresses associated with the adapters on the local computer.
  - IP_ADAPTER_ADDRESSES, which is the header node for a linked list of addresses for a particular adapter.
  - IP_ADAPTER_UNICAST_ADDRESS, which stores a single unicast IP address in a linked list of IP addresses for a particular adapter.
- Expanded the enumeration RTX64_MONITOR_CONTROL_OP to include operations for setting and resetting triggers for monitoring events (4139):
  - MONITOR_CONTROL_SET_EVENT_TRIGGERS – Deterministically sets triggers for monitoring events.
  - MONITOR_CONTROL_RESET_EVENT_TRIGGERS – Deterministically resets (i.e., turns off) triggers for monitoring events.
  - MONITOR_CONTROL_SET_CUSTOM_EVENT_TRIGGERS – Deterministically sets triggers for custom monitoring events.
  - MONITOR_CONTROL_RESET_CUSTOM_EVENT_TRIGGERS – Deterministically resets (i.e., turns off) triggers for custom monitoring events.
- Added Seek functionality to the Monitoring API. (3819)

Samples

- Added a new SimpleProducerConsumer sample that demonstrates performance impact caused by CPU cache. The sample builds two applications: a consumer and a producer. (2774)
- Added a new RTX64Config sample that contains a subset of the source code for RTX64Config, the command-line tool for configuration and control of the RTX64 Runtime component. This sample demonstrates how to use the RTX64 Managed Code Framework (IntervalZero.RTX64.dll), which provides programmatic access to all RTX64 configuration and control operations. (2834)

RTX64 2014 with Service Pack 1

Tools and Utilities

- Added support to log Windows custom events with support for RtGenerateEvent under Windows. (2972)
- Added optimizations to reduce generation of monitoring events to no more than 20 instructions in real-time code paths. (2870)

RT-TCP/IP Stack and Drivers

- Increased the size of datagram packets used by the RT-TCP/IP Stack from 8k to 64k. (3480)

- Added support for IGMPv3, MLDv2, and IPV6_MULTICAST_LOOP in the RT-TCP/IP Stack. (3573)
- Incorporated Intel 82579 controller support into this release of RTX64. (3436)

Debugging

- Added support to Visual Studio 2012 and 2013 for local and remote debugging of Real-time applications in Visual Studio:
    - Added local debugging support through launch. (1577, 3211)
    - Added support for Start without Debugging within the Visual Studio debugger. (2643)
    - Added support for launching a RTSS process on a remote target system for debugging. (3213)

SDK

- Added development support for Visual Studio 2013.
- Added new Visual Studio templates for creating a RTSS Application or Real-Time DLLs.
- Added the API call RtEnumProxyProcesses, which enumerates proxy processes associated with Windows processes linked to RTAPI.
- Added a collection of code snippets for Real-Time API calls that can be inserted where you need them in your code in Visual Studio.
- Added two new Winsock API calls:
    - inet_ntop – Converts an IPv4 or IPv6 Internet network address into a string in Internet standard format.
    - inet_pton – Converts an IPv4 or IPv6 Internet network address in its standard text presentation form into its numeric binary form.
- Added support for additional socket options in the RTX64 Winsock API. For the complete list of supported options, see getsockopt and setsockopt. (3554)

RTX64 2014

Subsystem

- Added monitoring infrastructure that allows you to profile real-time application behavior.
- Added AVX 2.0 support.
- Improved thread priority mapping between Windows processes and how they interact with the real-time subsystem. You now have complete control over how Window proxy threads interact with RTSS thread and objects.

Tools and Utilities

- Improved the control panel to allow for modification of Network interface friendly names.
- Improved error messages provided by tools when RTSS binaries are incorrectly stamped or user does not have the proper permissions.
- The control panel now provides the ability to disable the logic used by the RTX64 subsystem to prevent windows power management of processor speeds.
- Added new tools:
    - Task Manager – view active Real-time processes (.rtss) and windows processes linked to RTX64 (.exe). You can start new tasks and terminate currently running tasks.
    - Monitor – start and stop monitoring sessions, and generate log files of monitoring results.
- Analyzer now includes information about the status of Virtual Network components.

RT-TCP/IP Stack and Drivers

- Added a Virtual Network that provides a virtual point-to-point connection between Windows and RTX64. It emulates a local area network connection between Windows and the Real-time Subsystem with no additional hardware required.

Debugging

- Added the RTX64 WinDbg Extension, which extends Microsoft's 64-bit version of WinDbg and provides a way to analyze and interpret the state of RTSS applications and the RTX64 Subsystem.

SDK

- The managed code framework now provides functionality to disable the logic used by the RTX64 subsystem to prevent windows power management of processor speeds.
- The Managed code interface now provides functionality to allow Windows applications to enumerate RTSS processes.
- Added support for multiple RTX64 SDKs on the same system through common tools and versioned build environments.
- New RTAPI calls were added to support getting and setting of priorities for Windows proxy threads. New API calls are as follows:
    - RtSetProxyThreadPriority - sets the priority of a RTSS proxy thread from a Windows application.
    - RtGetProxyThreadPriority - gets the priority of a RTSS proxy thread from a Windows application.
    - RtkSetProxyThreadPriority - sets the priority of a RTSS proxy thread from a Windows kernel driver.
    - RtkGetProxyThreadPriority - gets the priority of a RTSS proxy thread from a Windows kernel driver.

- Added an API call RtGetEnabledXStateFeature which allows developers to determine the capabilities of the system processor.
- Added new API calls to support Monitoring functionality:
    - RtGenerateEvent – allows you to generate user defined events within an RTSS process.
    - RtMonitorControl – allows you to control monitoring within an RTSS process programmatically.
- Added new API calls to support link status monitoring:
    - RtnIsDeviceOnline – gets the online status of a network device for link status monitoring.
    - RtnIsStackOnline – gets the online status of the RT-TCP/IP stack for link status monitoring.
- Added support for the Windows API call GetModuleFileName, which retrieves a handle for each module in a specified process.
- Added support for the Windows API call TryEnterCriticalSection, which attempts to enter a critical section without blocking. If the call is successful, the calling thread takes ownership of the critical section.