Autorun 1.0

Quick Start Manual

The goal of Autorun is to provide a rapid development framework and a test/production environment for running automated tasks. The task scheduler can run interactively or as a Windows NT service and the monitoring interface can connect remotely to this service.

Table of Contents

Before you start…

What is Autorun ?

We often use to develop tasks that will be automated repeatedly. By experience, I noticed one particularity one that type of project: although the business function remains dramatically easy to develop, developers will spent most of their time in fixing and stabilizing the “automation” part of this project. Finally, the required functionality that should represent 80% of your project took 20% to setup although the system part (the remaining 20% of your project) of your program took 80%...

Moreover developing Windows NT Service is a real nightmare to debug: newbies in that domain will have certainly tried to attach to a process… Although there are best practices to debug Windows services, these are not always fast enough.

The goal of Autorun is to provide a rapid development framework and a test/production environment for running automated tasks. The task scheduler can run interactively or as a Windows NT service and the monitoring interface can connect remotely to this service.

How does it work?

Autorun can be used in several forms:

- As an Windows NT service (start “autorun.exe”)
- As an interactive Windows application (start “autorun.exe /server”)
- Built-in in the client configuration tool (start “autorun.exe /client”)

Installing Autorun

The system requirements to use Autorun are:

- Microsoft Windows XP or above
- Microsoft Framework.NET 2.0
- Microsoft SQL Server Express minimum

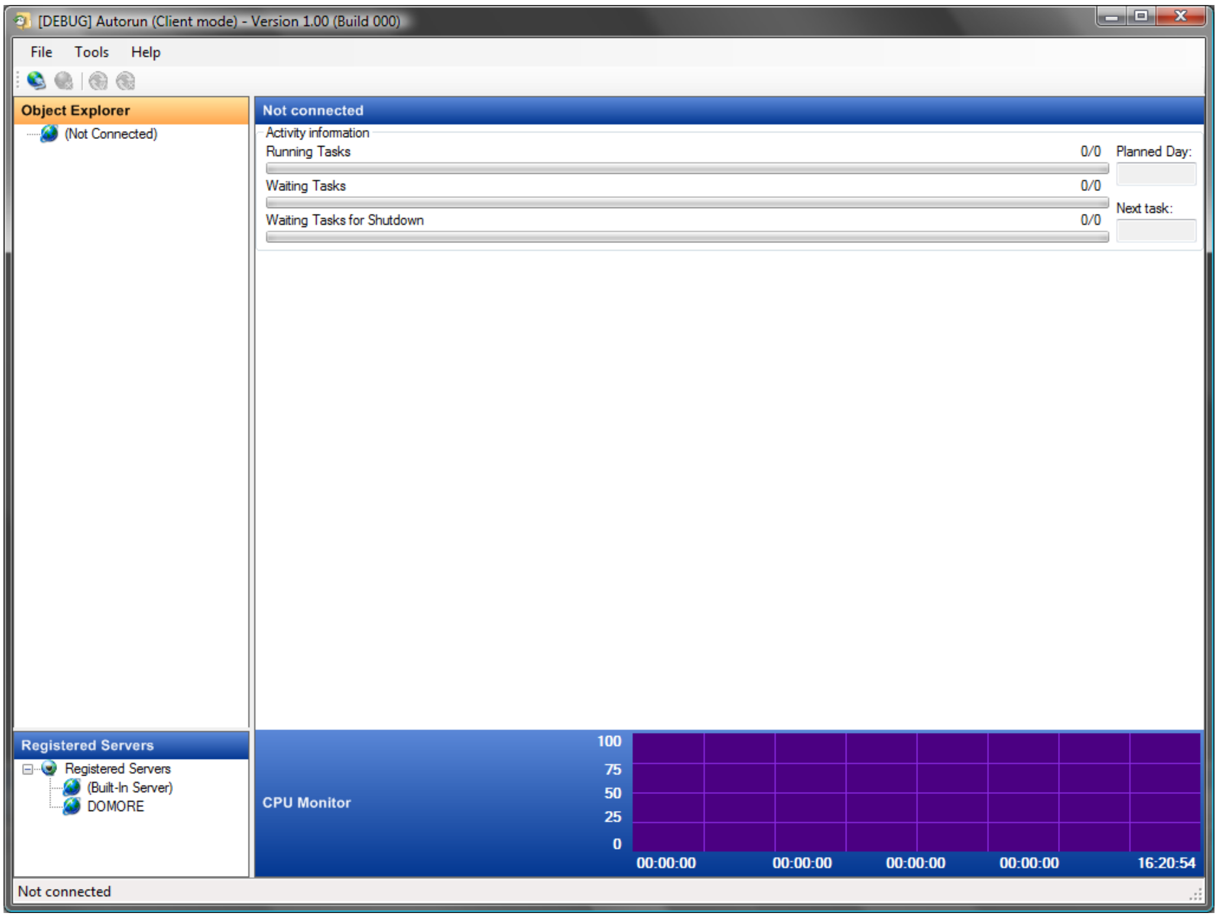
1. Copy the core files in the desired directory:
   - Autorun.exe
   - Autorun.exe.config
   - Autorun.Framework.dll
   - Autorun.mdf
   - Autorun_log.LDF
2. Optionally, attach the database to a SQL Server instance
3. Edit the system configuration file
   a. Change the database connection string
4. Optionally create 2 shortcuts
   a. A shortcut “Autorun (Client)” pointing to “Autorun.exe /client”
   b. A shortcut “Autorun (Server)” pointing to “Autorun.exe /server”

Quick start guide

Start Autorun in built-in mode

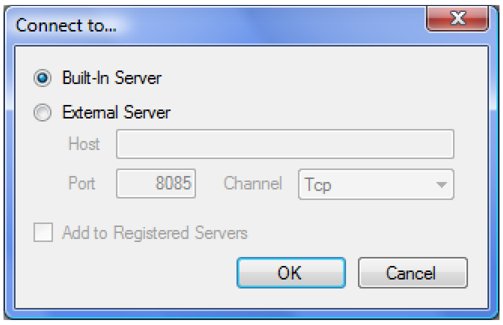
In this mode, the instance of Autorun is integrated in the server configuration tool.

- Start "[Home]\autorun.exe" /client



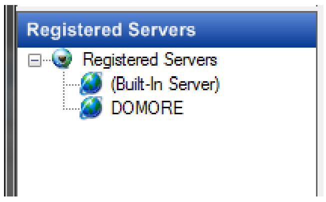
Connect to the server

To connect to a server, select the menu entry "File / Connect" (or Ctrl+C). The connect dialog box appears:

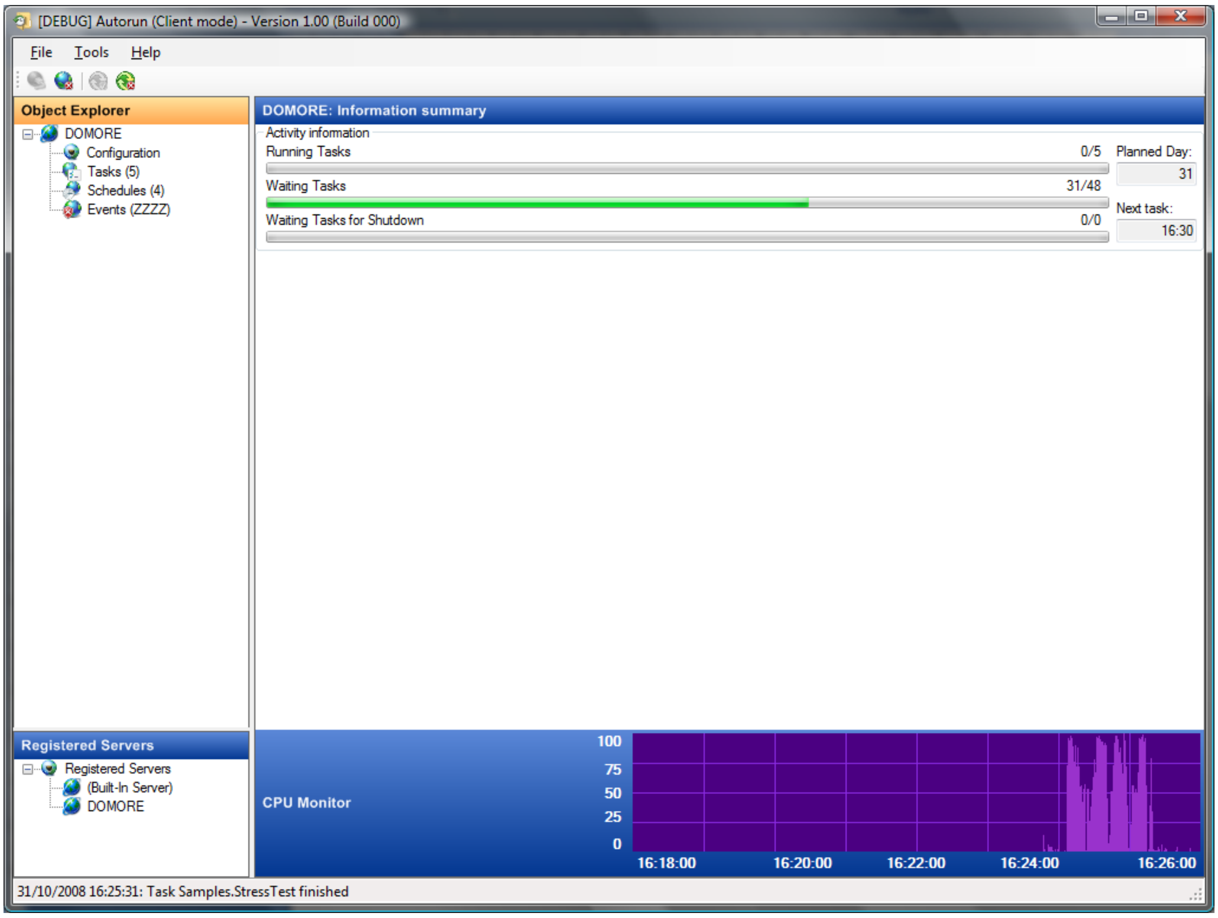


Select "Built-in" server and press the "ok" button.

- You can also directly connect to the internal server by double-clicking on the item "(Built-In Server)" in the "Registered Servers" pane.



When successfully connected, you can browser the server with the object explorer:



31/10/2008 16:25:31: Task Samples.StressTest finished

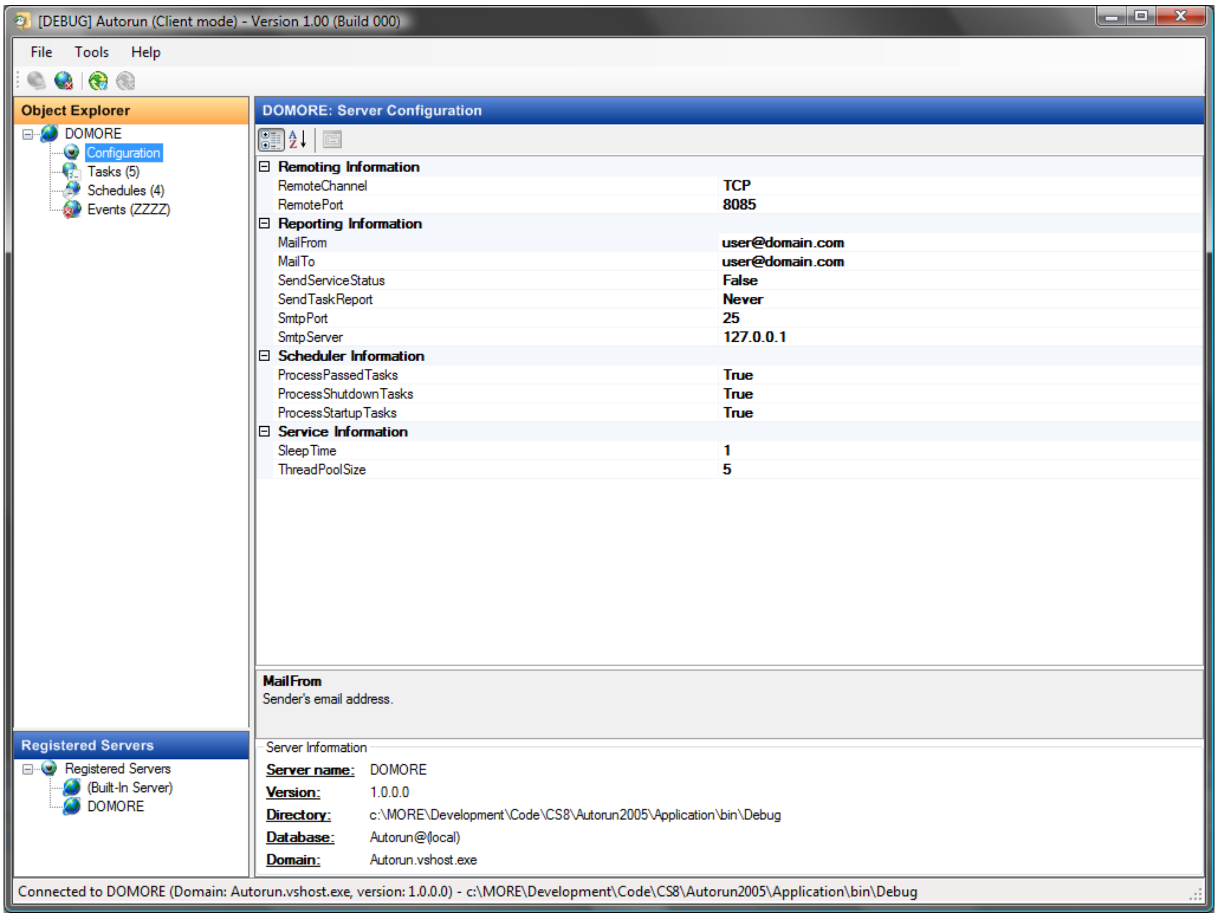
The information summary

The "information summary" pane displays general information on the server:

- The number of tasks currently active
- The number of tasks waiting to be executed until the end of the day
- The number of tasks waiting to be executed before the server has been stopped
- The processor resources consumed by Autorun

Configuring the server

As soon as you are connected to the server, you can view and edit its configuration settings remotely through the client configuration tool.



```
Remoting Information
    RemoteChannel          TCP
    RemotePort             8085
Reporting Information
    MailFrom               user@domain.com
    MailTo                 user@domain.com
    SendServiceStatus      False
    SendTaskReport         Never
    SmtpPort               25
    SmtpServer             127.0.0.1
Scheduler Information
    ProcessPassedTasks     True
    ProcessShutdownTasks   True
    ProcessStartupTasks    True
Service Information
    SleepTime              1
    ThreadPoolSize         5

MailFrom
Sender's email address.

Server Information
Server name:   DOMORE
Version:       1.0.0.0
Directory:     c:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug
Database:      Autorun@(local)
Domain:        Autorun.vshost.exe

Connected to DOMORE (Domain: Autorun.vshost.exe, version: 1.0.0.0) - c:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug
```

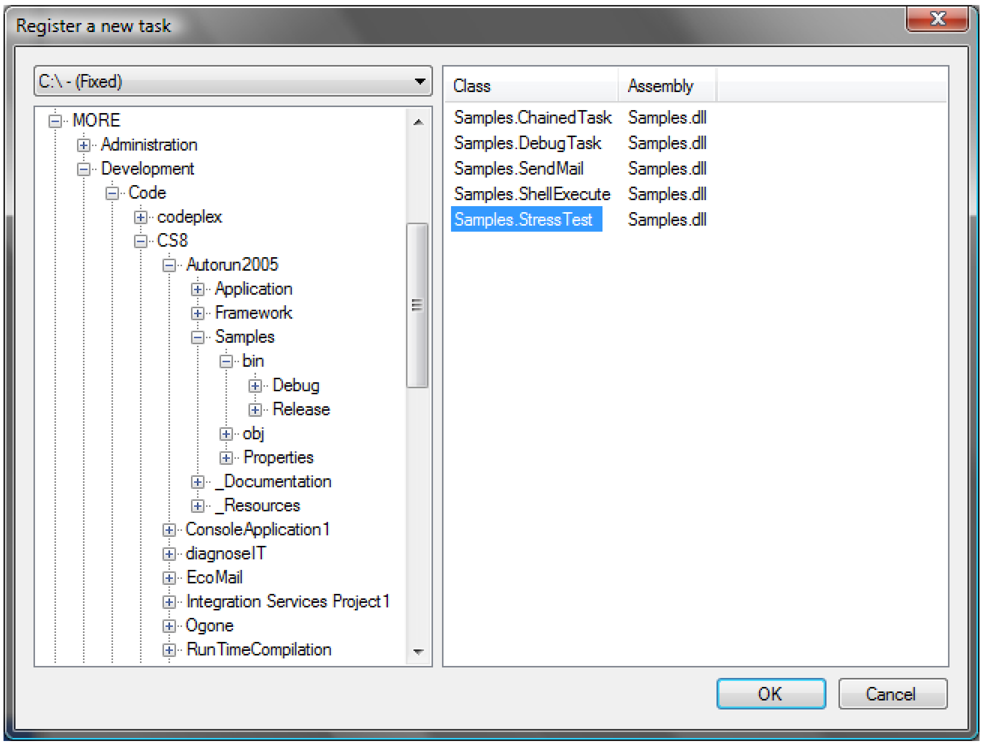
Remote Channel	Protocol to be used to remotely connect to the server. By default, the protocol used is tcp. *When changing this value, you need to restart the server.*
Remote Port	Port used to remotely connect to the server. The default value is 8085. *When changing this value, you need to restart the server.*
Mail From	Sender’s email address
Mail To	Receiver’s email address
SMTP Port	Port of the SMTP server
SMTP host	Host name of the SMTP server
SMTP Delivery Method	Specify how to send email: through the network, through the standard IIS Pickup directory, through a custom Pickup directory
SMTP Pickup Directory	Location of the custom Pickup directory
Send Service Status	Should Autorun send an email every time the server has been started and stopped?
Send Task Reports	Define when task reports are sent: never, when a tasks encounters an error only, or always
Send Daily Reports	Should Autorun send a daily activity report summarizing all tasks that have run and their exit codes? Should it send a daily report even if no task has run?
Process Startup Tasks	When the server is started, should startup task be planned?
Process Shutdown Tasks	When the server is started, should shutdown tasks be planned?
Process Passed Tasks	When the server is started, should Autorun plan the tasks that should have been executed earlier this day?
Sleep Time	When the server is idle (waiting for tasks to end, or new task to be started),

	define the period it release the processor resources
Thread Pool Size	Number of tasks the server can run concurrently

Registering tasks

For security reasons, Autorun permits the execution of registered tasks only. Even if a task starts another task, the chained task must be registered.

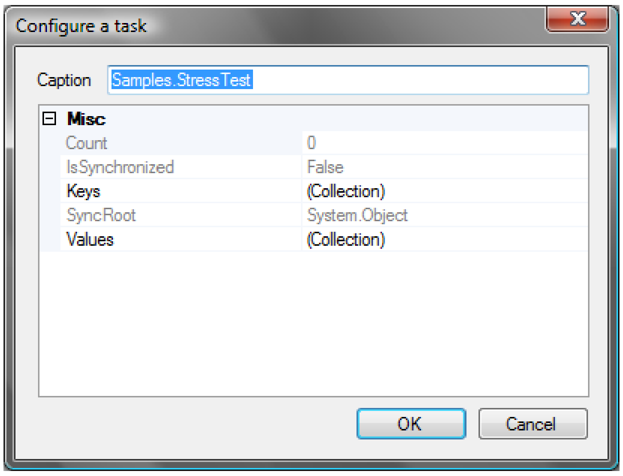
In this quick start, we will register tasks provided in the sample library. To create a new task entry, right click on the empty task list and click on the "New" menu entry (or press the key Alt+N). The file browser dialog box appears. This dialog box presents all the drives accessible by the remote server.



Class	Assembly
Samples.ChainedTask	Samples.dll
Samples.DebugTask	Samples.dll
Samples.SendMail	Samples.dll
Samples.ShellExecute	Samples.dll
Samples.StressTest	Samples.dll

Locate the task "StressTest" in the Samples library and click on "OK".

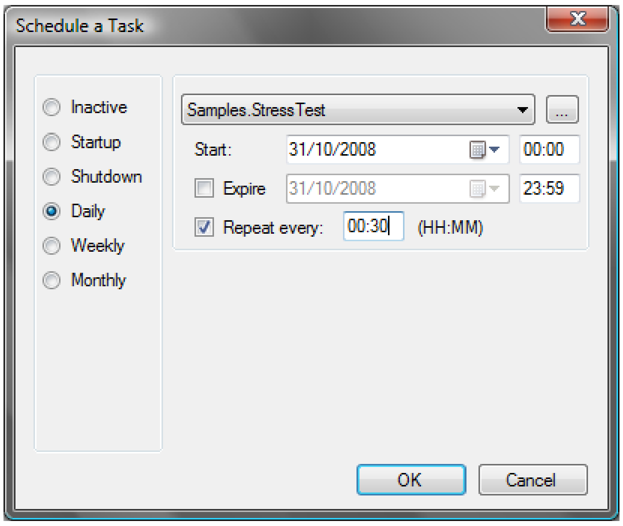
The next dialog box lets you specify a friendly name to the task and edit the configuration settings of the tasks, if any.



Caption: Samples.StressTest

Misc	
Count	0
IsSynchronized	False
Keys	(Collection)
SyncRoot	System.Object
Values	(Collection)

The task StressTest does not present any configuration setting. Click on “OK” to continue. The Latest dialog box lets you configure a first schedule with this task.



Samples.StressTest

Start: 31/10/2008 00:00

[ ] Expire 31/10/2008 23:59

[✓] Repeat every: 00:30 (HH:MM)

( ) Inactive ( ) Startup ( ) Shutdown [✓ Daily] ( ) Weekly ( ) Monthly

In this tutorial, let’s schedule this task daily, repeating every 30 minutes. Click on “OK” when finished.

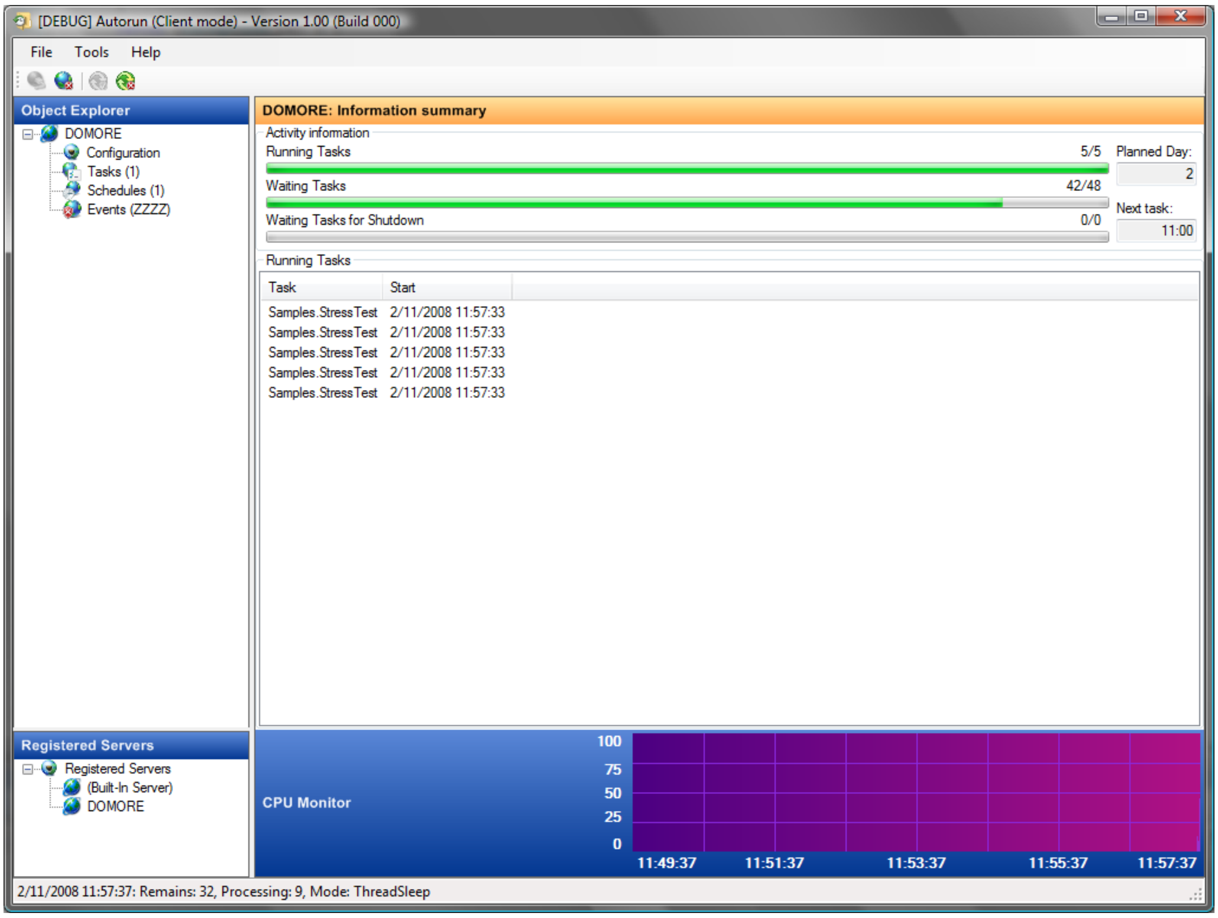
Starting the server

Now that you have registered your tasks and configure schedules, it's time to start Autorun! You can start it the following ways:

- Select the menu entry "Tools / Start"
- Icon "Start" on the toolbar
- Shortcut key Ctrl+F5

The "information summary" pane lets you control the activity of the server:

- How many tasks are currently running?
- How many tasks are in the waiting queues?
- When should the next waiting task be started?
- Which day has been last planned?



DOMORE: Information summary

Activity information

Running Tasks 5/5
Waiting Tasks 42/48
Waiting Tasks for Shutdown 0/0
Planned Day: 2
Next task: 11:00

Running Tasks

Task	Start
Samples.StressTest	2/11/2008 11:57:33
Samples.StressTest	2/11/2008 11:57:33
Samples.StressTest	2/11/2008 11:57:33
Samples.StressTest	2/11/2008 11:57:33
Samples.StressTest	2/11/2008 11:57:33

2/11/2008 11:57:37: Remains: 32, Processing: 9, Mode: ThreadSleep

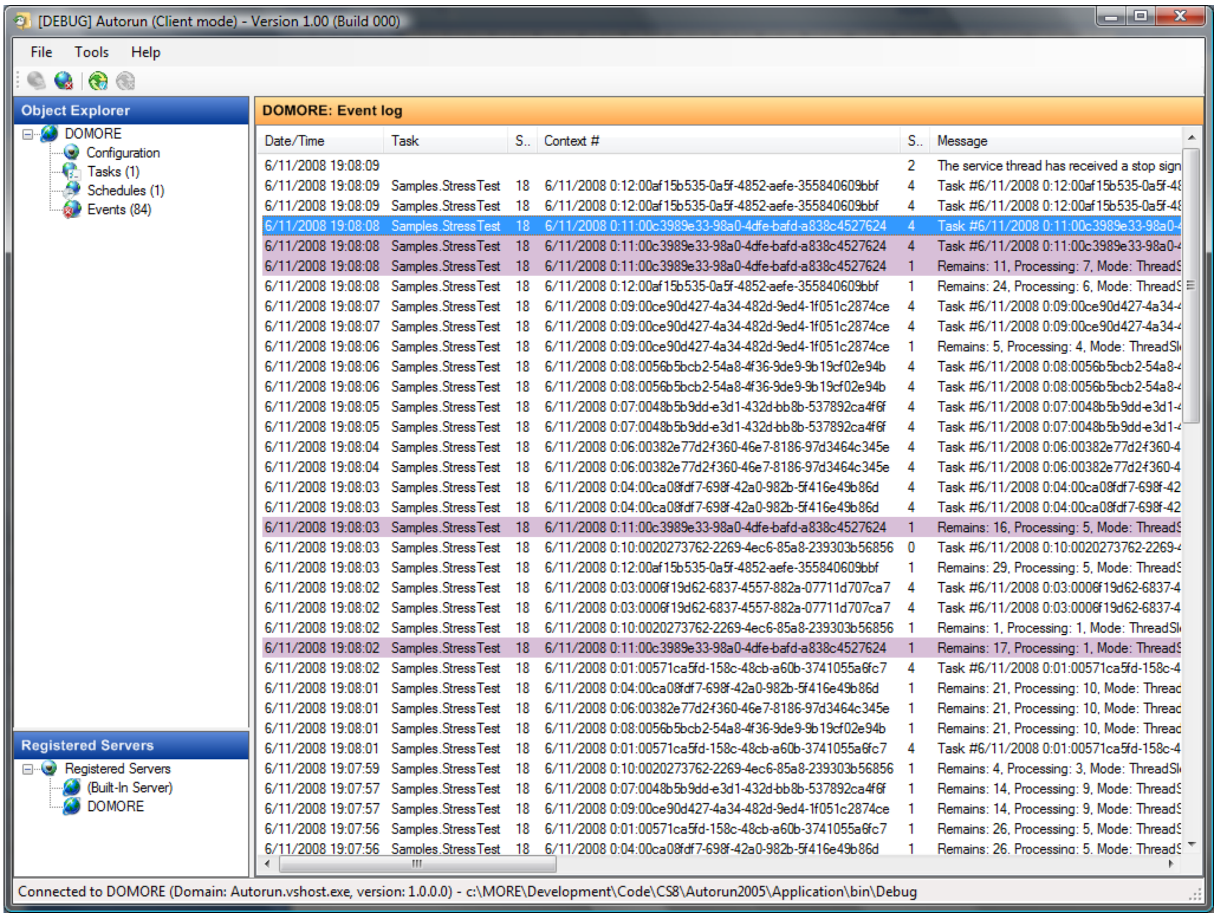
Notes:

➢ The task list is not automatically refreshed: to update its content, right click and select the menu entry "Refresh"

Monitoring tasks activity

When tasks are executed, they can send notifications to Autorun. You can access these events through the event pane that is illustrated below.

When you click on an event entry, other lines are highlighted in light purple: these events are related to the same execution context.



Date/Time	Task	S..	Context #	S..	Message
6/11/2008 19:08:09				2	The service thread has received a stop sign
6/11/2008 19:08:09	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	4	Task #6/11/2008 0:12:00af15b535-0a5f-48
6/11/2008 19:08:09	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	4	Task #6/11/2008 0:12:00af15b535-0a5f-48
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	4	Task #6/11/2008 0:11:00c3989e33-98a0-
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	4	Task #6/11/2008 0:11:00c3989e33-98a0-
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	1	Remains: 11, Processing: 7, Mode: Thread
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	1	Remains: 24, Processing: 6, Mode: Thread
6/11/2008 19:08:07	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	4	Task #6/11/2008 0:09:00ce90d427-4a34-
6/11/2008 19:08:07	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	4	Task #6/11/2008 0:09:00ce90d427-4a34-
6/11/2008 19:08:06	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	1	Remains: 5, Processing: 4, Mode: ThreadSl
6/11/2008 19:08:06	Samples.StressTest	18	6/11/2008 0:08:0056b5bcb2-54a8-4f36-9de9-9b19cf02e94b	4	Task #6/11/2008 0:08:0056b5bcb2-54a8-
6/11/2008 19:08:06	Samples.StressTest	18	6/11/2008 0:08:0056b5bcb2-54a8-4f36-9de9-9b19cf02e94b	4	Task #6/11/2008 0:08:0056b5bcb2-54a8-
6/11/2008 19:08:05	Samples.StressTest	18	6/11/2008 0:07:0048b5b9dd-e3d1-432d-bb8b-537892ca4f6f	4	Task #6/11/2008 0:07:0048b5b9dd-e3d1-
6/11/2008 19:08:05	Samples.StressTest	18	6/11/2008 0:07:0048b5b9dd-e3d1-432d-bb8b-537892ca4f6f	4	Task #6/11/2008 0:07:0048b5b9dd-e3d1-
6/11/2008 19:08:04	Samples.StressTest	18	6/11/2008 0:06:00382e77d2-f360-46e7-8186-97d3464c345e	4	Task #6/11/2008 0:06:00382e77d2-f360-4
6/11/2008 19:08:04	Samples.StressTest	18	6/11/2008 0:06:00382e77d2-f360-46e7-8186-97d3464c345e	4	Task #6/11/2008 0:06:00382e77d2-f360-4
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	4	Task #6/11/2008 0:04:00ca08fdf7-698f-42
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	4	Task #6/11/2008 0:04:00ca08fdf7-698f-42
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	1	Remains: 16, Processing: 5, Mode: Thread
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:10:0020273762-2269-4ec6-85a8-239303b56856	0	Task #6/11/2008 0:10:0020273762-2269-
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	1	Remains: 29, Processing: 5, Mode: Thread
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:03:0006f19d62-6837-4557-882a-07711d707ca7	4	Task #6/11/2008 0:03:0006f19d62-6837-4
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:03:0006f19d62-6837-4557-882a-07711d707ca7	4	Task #6/11/2008 0:03:0006f19d62-6837-4
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:10:0020273762-2269-4ec6-85a8-239303b56856	1	Remains: 1, Processing: 1, Mode: ThreadSl
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	1	Remains: 17, Processing: 1, Mode: Thread
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:01:00571ca5fd-158c-48cb-a60b-3741055a6fc7	4	Task #6/11/2008 0:01:00571ca5fd-158c-4
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	1	Remains: 21, Processing: 10, Mode: Threac
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:06:00382e77d2-f360-46e7-8186-97d3464c345e	1	Remains: 21, Processing: 10, Mode: Threac
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:08:0056b5bcb2-54a8-4f36-9de9-9b19cf02e94b	1	Remains: 21, Processing: 10, Mode: Threac
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:01:00571ca5fd-158c-48cb-a60b-3741055a6fc7	4	Task #6/11/2008 0:01:00571ca5fd-158c-4
6/11/2008 19:07:59	Samples.StressTest	18	6/11/2008 0:10:0020273762-2269-4ec6-85a8-239303b56856	1	Remains: 4, Processing: 3, Mode: ThreadSl
6/11/2008 19:07:57	Samples.StressTest	18	6/11/2008 0:07:0048b5b9dd-e3d1-432d-bb8b-537892ca4f6f	1	Remains: 14, Processing: 9, Mode: Thread
6/11/2008 19:07:57	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	1	Remains: 14, Processing: 9, Mode: Thread
6/11/2008 19:07:56	Samples.StressTest	18	6/11/2008 0:01:00571ca5fd-158c-48cb-a60b-3741055a6fc7	1	Remains: 26, Processing: 5, Mode: Thread
6/11/2008 19:07:56	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	1	Remains: 26, Processing: 5, Mode: Thread

Connected to DOMORE (Domain: Autorun.vshost.exe, version: 1.0.0.0) - c:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug

Killing a task

Rarely, it might happen that a task is not working well! When this unbelievable scenario occurs, the last action you have to do is to kill the task and release the resources on the server:

- First, refresh the list of running tasks: right click on the grid and select the “Refresh” menu item.
- Then right click on the selected task and select the menu entry “Kill this task”

The task will receive a stop signal and will be stopped after a delay of one second.

Stopping the server

In some case, you might need to force the server to stop. You can stop it the following ways:

- Select the menu entry “Tools / Stop”
- Icon “Stop” on the toolbar
- Shortcut key Ctrl+F8

When stopping the server, Autorun tries to stop gently the running tasks during 30 seconds. During that same period, shutdown tasks are started also. After this period, all remaining tasks are killed.

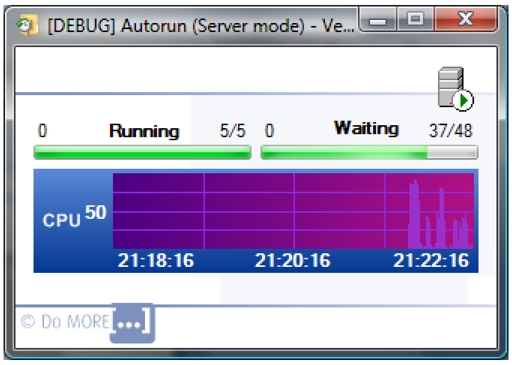
Notes:

- When a task receives a stop signal, it waits for 1 second to achieve; after this period, the task kills itself.
- Be careful when forcing Autorun to stop; stopping running tasks might leave your systems inconsistent.

Start Autorun as an interactive Windows Server

In this mode, the instance of Autorun is started in its own standalone interactive Windows process. This mode is convenient when you want to fine tune the configuration on the target server.

- Start "[Home]\autorun.exe" /server

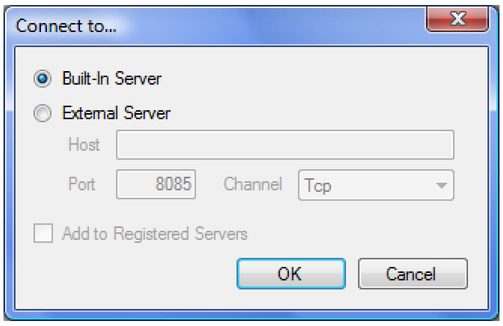


To pilot the server, you need to use the client configuration tool.

Connecting to the server

To remotely manage Autorun, you must use the client configuration tool.

- Start "[Home]\autorun.exe" /client
- Select the menu entry "File / Connect" (or Ctrl+C). The connect dialog box appears:

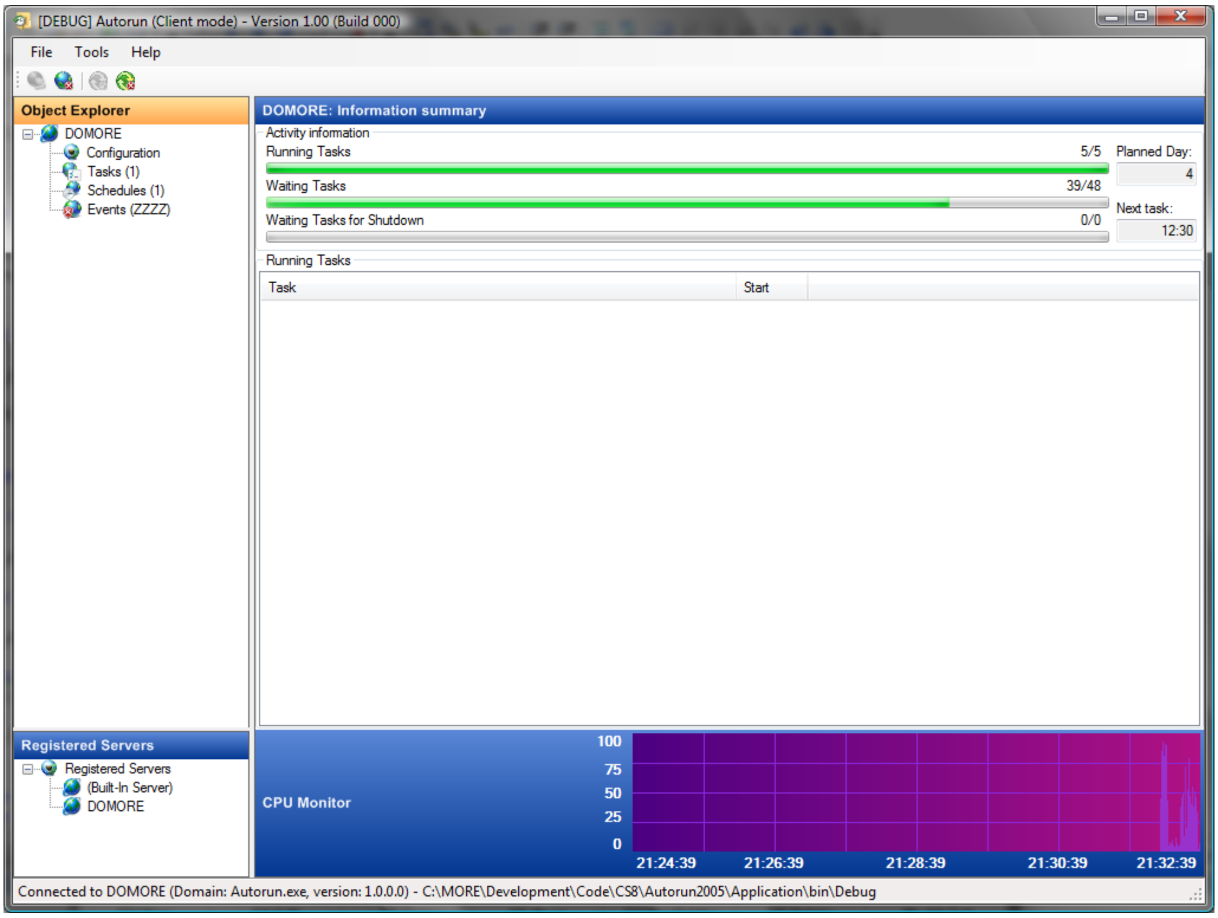

Connect to...

(●) Built-In Server
( ) External Server
Host
Port 8085 Channel Tcp
[ ] Add to Registered Servers
OK Cancel

- Select the "External Server" mode
- Put the computer name or its IP address in the "Host" field
- Specify the port and the channel, or leave the default values if you did not change them
- If you want to add this connection setting, check the option "Add to Registered Servers"
- Press "OK"

Note about registered servers:

When checking the option “Add to Registered Servers”, you add the connection settings to the “Registered Server” pane on the main form. Later, you can quickly connect to a server by double clicking on the server name.



Connected to DOMORE (Domain: Autorun.exe, version: 1.0.0.0) - C:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug

Note about refresh rate of the “information summary” pane

When connected remotely, the client configuration tool cannot receive events from the server asynchronously. Therefore, an acceptable refresh rate is done by refreshing every second the information summary pane.

However, notifications coming from tasks are not displayed interactively.

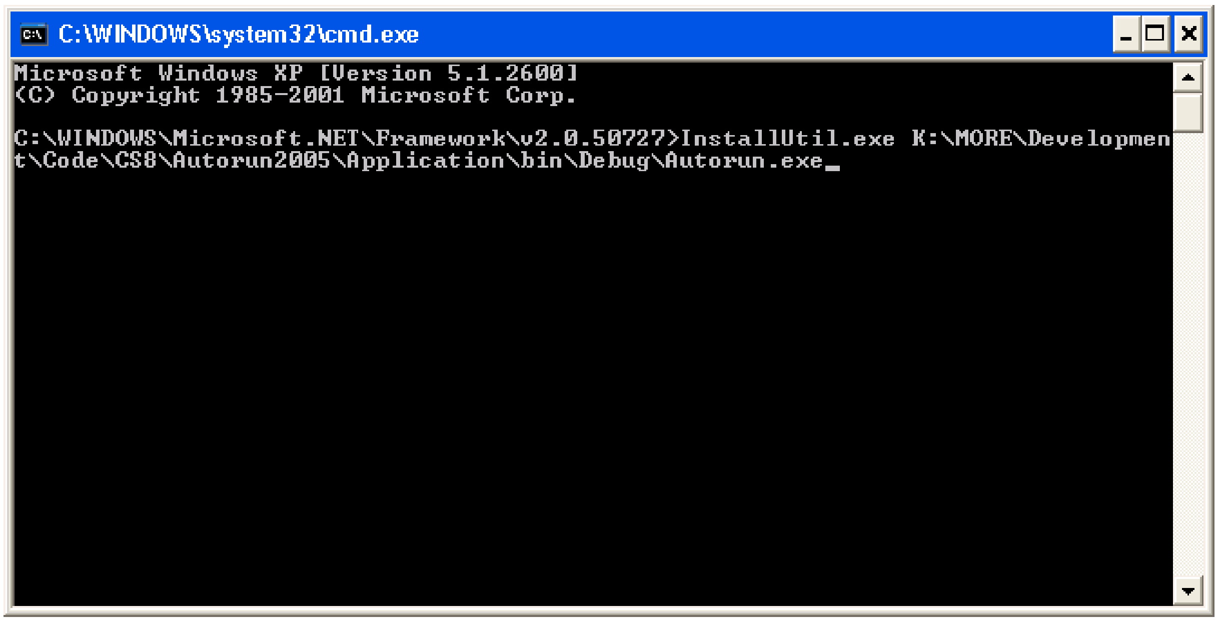
Start Autorun as a Windows Service

In a production environment, it is not necessary to execute Autorun interactively. In this case, starting Autorun as a Windows service is convenient.

Installing Autorun as a Windows Service

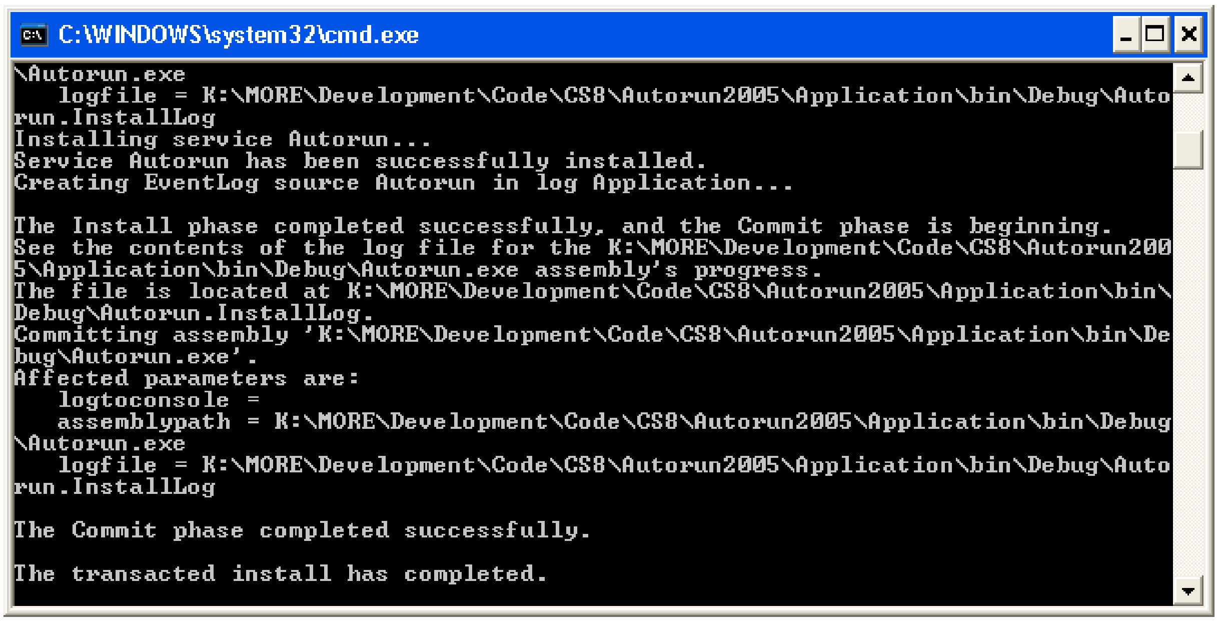
The first step is to register Autorun as a service. You can do it by using the tool installutil.exe provided with the Microsoft Framework.NET 2.0.

- Start a command prompt
- Go to C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727
- Type the command "installutil.exe [Home]\Autorun.exe" (where [Home] is the directory where Autorun is located."



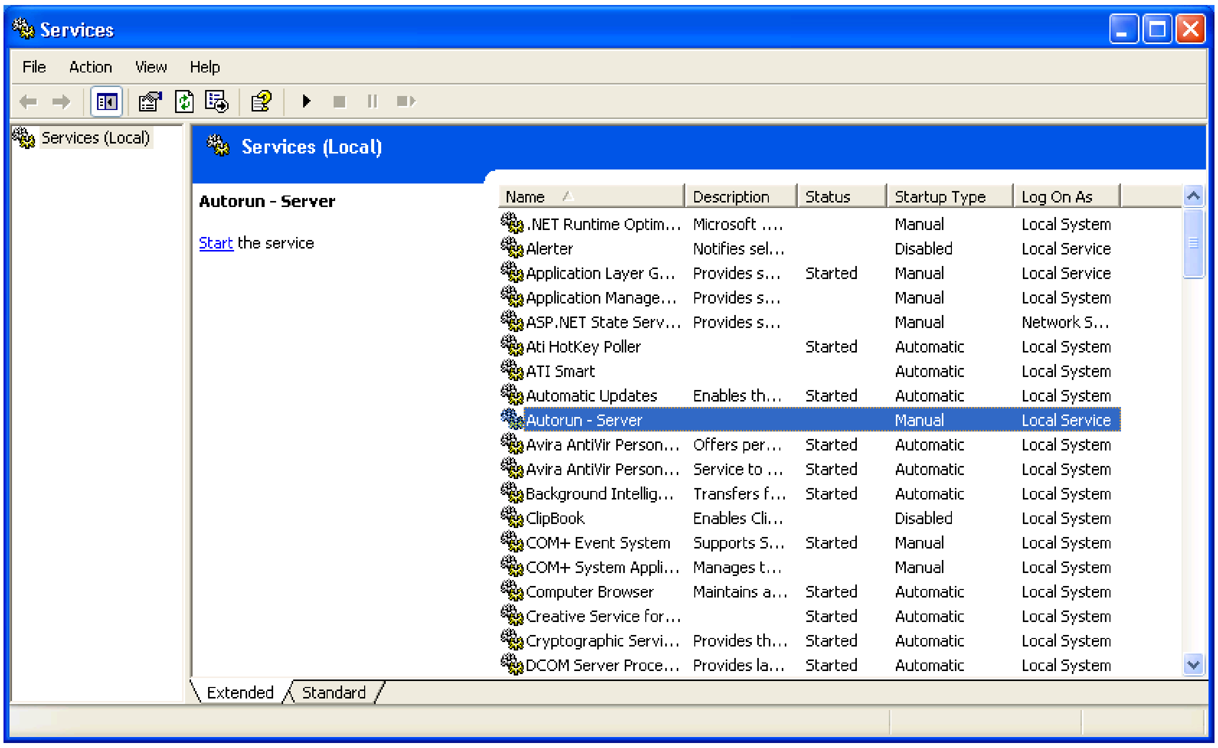
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>InstallUtil.exe K:\MORE\Developmen
t\Code\CS8\Autorun2005\Application\bin\Debug\Autorun.exe_
```

- A successful install will result in the message "The Commit phase completed successfully"



```
\Autorun.exe
   logfile = K:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug\Auto
run.InstallLog
Installing service Autorun...
Service Autorun has been successfully installed.
Creating EventLog source Autorun in log Application...

The Install phase completed successfully, and the Commit phase is beginning.
See the contents of the log file for the K:\MORE\Development\Code\CS8\Autorun200
5\Application\bin\Debug\Autorun.exe assembly's progress.
The file is located at K:\MORE\Development\Code\CS8\Autorun2005\Application\bin\
Debug\Autorun.InstallLog.
Committing assembly 'K:\MORE\Development\Code\CS8\Autorun2005\Application\bin\De
bug\Autorun.exe'.
Affected parameters are:
   logtoconsole =
   assemblypath = K:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug
\Autorun.exe
   logfile = K:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug\Auto
run.InstallLog

The Commit phase completed successfully.

The transacted install has completed.
```

- Start the Services Management located in the configuration panel, administrative tools



Autorun - Server

Start the service

Name	Description	Status	Startup Type	Log On As
.NET Runtime Optim...	Microsoft		Manual	Local System
Alerter	Notifies sel...		Disabled	Local Service
Application Layer G...	Provides s...	Started	Manual	Local Service
Application Manage...	Provides s...		Manual	Local System
ASP.NET State Serv...	Provides s...		Manual	Network S...
Ati HotKey Poller		Started	Automatic	Local System
ATI Smart			Automatic	Local System
Automatic Updates	Enables th...	Started	Automatic	Local System
Autorun - Server			Manual	Local Service
Avira AntiVir Person...	Offers per...	Started	Automatic	Local System
Avira AntiVir Person...	Service to ...	Started	Automatic	Local System
Background Intellig...	Transfers f...	Started	Automatic	Local System
ClipBook	Enables Cli...		Disabled	Local System
COM+ Event System	Supports S...	Started	Manual	Local System
COM+ System Appli...	Manages t...		Manual	Local System
Computer Browser	Maintains a...	Started	Automatic	Local System
Creative Service for...		Started	Automatic	Local System
Cryptographic Servi...	Provides th...	Started	Automatic	Local System
DCOM Server Proce...	Provides la...	Started	Automatic	Local System

- Locate Autorun
  - Configure it to start automatically if you want
  - Start it

Tip

Before executing Autorun as a Windows service, run it as an interactive application first: it will be easier to control the configuration settings are correct (i.e.: database connection string).

Note about installing Autorun as a Windows service on Windows Vista

At this time, I did not figure out how to install successfully the service on Vista: there is a problem with the event log ("Security" log was not found)

Uninstalling Autorun

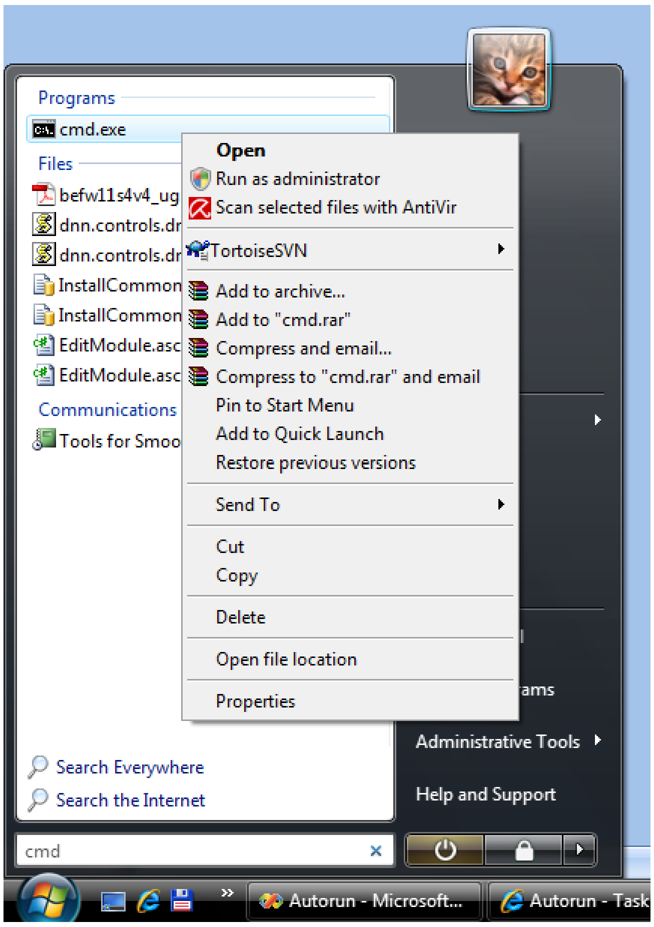
To unregister Autorun from the Windows services, simply use the same tools installutil.exe provided with the Microsoft Framework.NET.

- Type the command "installutil.exe /u [Home]\Autorun.exe" (where [Home] is the directory where Autorun is located."

Notes for Windows Vista users

Since the User Access Control (the well-known UAC) component restricts administrator privileges, you need to run the command prompt as an administrator:

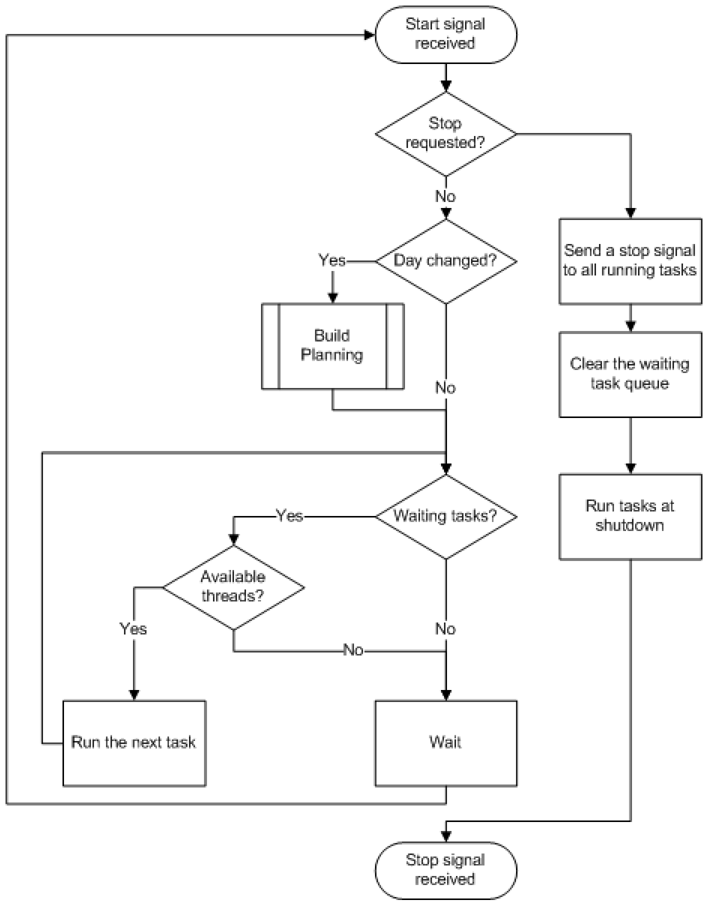
- In the start menu, type “cmd” as the search criteria
- Right click on cmd.exe and select the menu item “Run as administrator”



Administrator's guide

The lifecycle of Autorun

As the system administrator it is important to understand how Autorun works. The following figure illustrates the main loop of Autorun when it has been started:



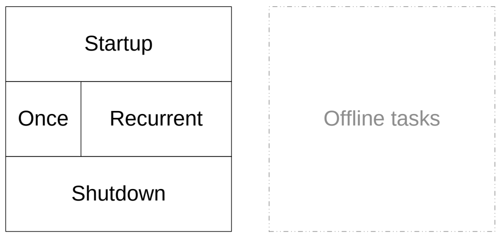
When started, Autorun executes tasks each time a free thread is available: it does not exceed the number of threads defined in the configuration. When all threads are used, Autorun goes into sleep mode, that is, it releases the processor resource for a given amount of time (set by the parameter "SleepTime").

When the day changed, the scheduler plans new tasks to be run on the scheduled day: when the scheduler runs the first time since the start signal, it also schedules tasks to run at shutdown. Planning a task consists of creating a task context: the service will load the task object only when it will run it.

When a stop signal is received, Autorun has 30 seconds to flush all active tasks and execute all tasks to be executed when it shuts down. When it sends a stop signal to a task, this task has 1 second to stop itself, otherwise it is aborted.

Scheduling tasks

When a task is scheduled, it must be started in one of the four following modes:



When	Description
Startup	These tasks are executed when the service starts. For example, all tasks that must be run when your computer has been powered on.
Once	These tasks are run once on a specific date and time
Recurrent	These tasks are recurring tasks; their schedule configuration determines when they are executed.
Shutdown	These tasks are executed when the service stop. For example, all tasks that must be run when your computer shuts down

As you can see on the above figure, there is a block named "offline tasks". For security purposes, all tasks that are indirectly executed through task chaining must be known (registered) by the system: a task is at least known as an "offline task". In debugging mode, task should not be registered in order to speed up development and test processes.

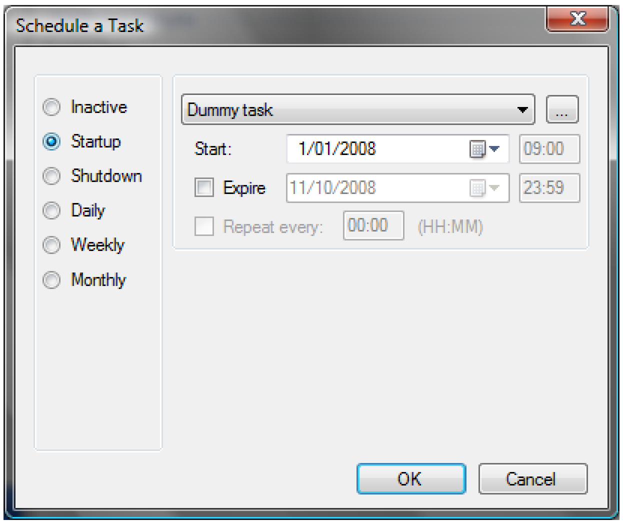
Although startup, once and shutdown tasks are quite easy to plan; recurring tasks are more complex to schedule.

Depending on the start mode, some parameters might be required. The following figure illustrates the parameter requirement:

Start mode	From time	To time	Step	Days of week
None	*Not used*	*Not used*	*Not used*	*Not used*
Startup	*Not used*	*Not used*	*Not used*	*Not used*
Once	Run time	*Not used*	*Not used*	Allowed days
Recurring	First run time	Last allowed run time	Run interval	Allowed days
Shutdown	*Not used*	*Not used*	*Not used*	*Not used*

Startup tasks

These tasks are executed when the Server is started. Startup tasks can be limited to a period of time.



Schedule a Task

Inactive [✓ Startup] Shutdown Daily Weekly Monthly

Dummy task

Start: 1/01/2008 09:00

Expire 11/10/2008 23:59

Repeat every: 00:00 (HH:MM)

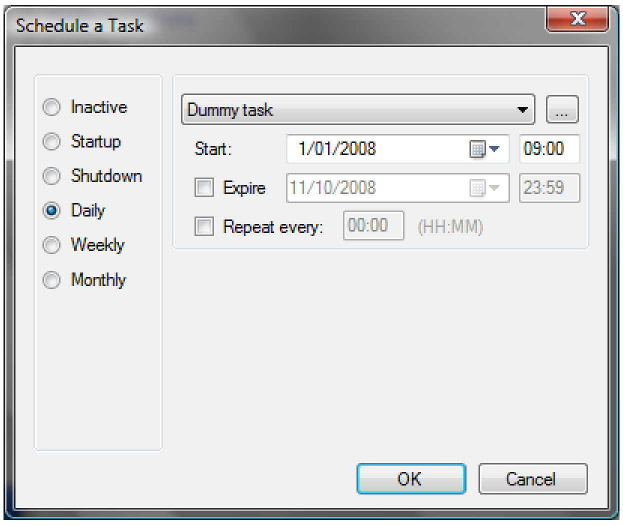
OK Cancel

Daily schedule

A daily schedule represents a task that runs every day in a contiguous period. This task can run many times during a day: when the day changes, the number a occurrences to be executed is computed again. For example, if a task is scheduled from 1st Jan 10.00am to 2nd Jan 3.00am every 3 jours, the occurrences will be:

- 1st Jan: 10.00am, 1.00pm, 4.00pm, 7.00pm, 10.00pm
- 2nd Jan: 0.00am, 3.00am, 6.00am, 9.00am

This is due to the design of the schedule manager: it plans an entire day, day by day without taking into account preceding days.



Schedule a Task

Inactive Startup Shutdown [✓ Daily] Weekly Monthly

Dummy task

Start: 1/01/2008 09:00

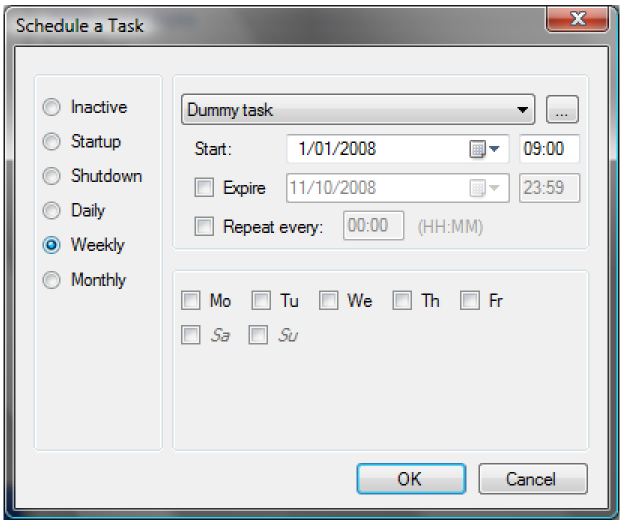
Expire 11/10/2008 23:59

Repeat every: 00:00 (HH:MM)

OK Cancel

Weekly schedule

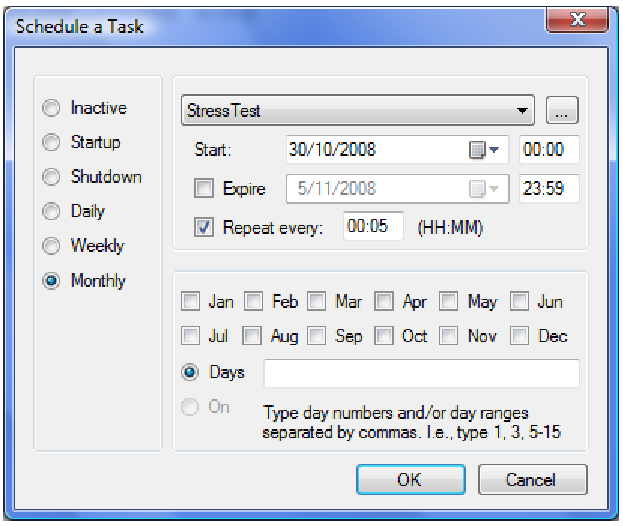
A weekly schedule is useful when a task cannot run on specific week days.



Weekly selected. Task: Dummy task. Start: 1/01/2008 09:00. Expire: 11/10/2008 23:59 (unchecked). Repeat every: 00:00 (HH:MM) (unchecked). Days: Mo, Tu, We, Th, Fr, Sa, Su (all unchecked).

Monthly schedule

With monthly schedule, you can decide which months the task must run. Moreover, you can specify if a task is to be executed on specific days of the month, or relative days (i.e.: first Monday)

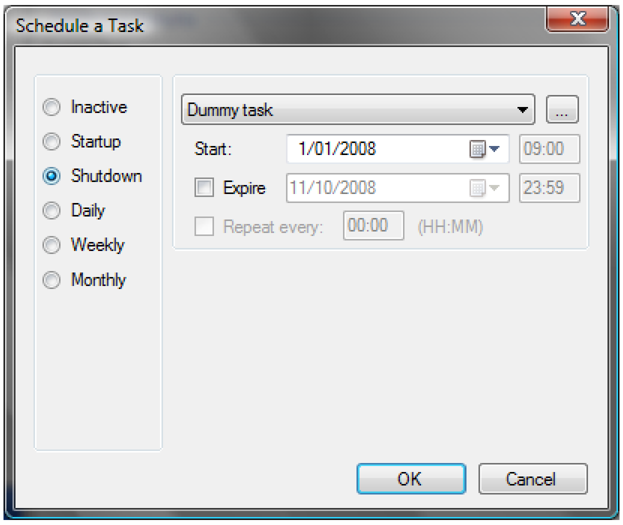


Monthly selected. Task: StressTest. Start: 30/10/2008 00:00. Expire: 5/11/2008 23:59 (unchecked). Repeat every: 00:05 (HH:MM) (checked). Months: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec (all unchecked). [✓ Days] On

Type day numbers and/or day ranges separated by commas. I.e., type 1, 3, 5-15

Shutdown tasks

These tasks are executed when the server is prompted to shutdown. Although it is not a good practice to start tasks when shutting down the server, this can be convenient to run small tasks, such as sending a notification to inform the server has shut down, …


Schedule a Task

Inactive ○ Startup ○ [✓ Shutdown] ○ Daily ○ Weekly ○ Monthly

Dummy task

Start: 1/01/2008 09:00

Expire 11/10/2008 23:59

Repeat every: 00:00 (HH:MM)

Monitoring the activity

Autorun provides several manners to monitor its activity:

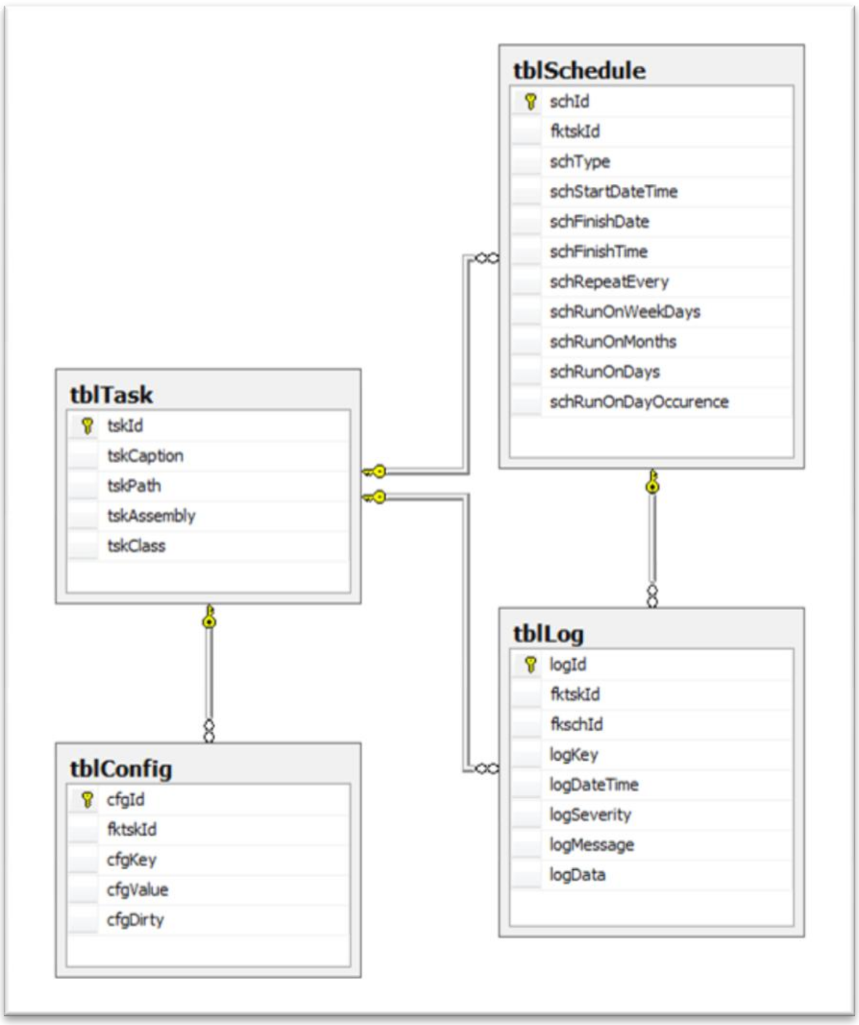
- Notifications sent by email when the service status has changed
- Tasks reports sent by email
- Daily activity report
- Log table in the server database

Before receiving emails from Autorun, be sure you configured correctly the mail settings. These settings are:

Mail From	Sender's email address
Mail To	Receiver's email address
SMTP Port	Port of the SMTP server (by default: 25)
SMTP host	Host name of the SMTP server (by default: 127.0.0.1)
SMTP Delivery Method	Send method (by default: Network)
SMTP Pickup Directory	Custom Pickup directory location, when delivery method is SpecifiedPickupDirectory (by default: nothing)

The log table

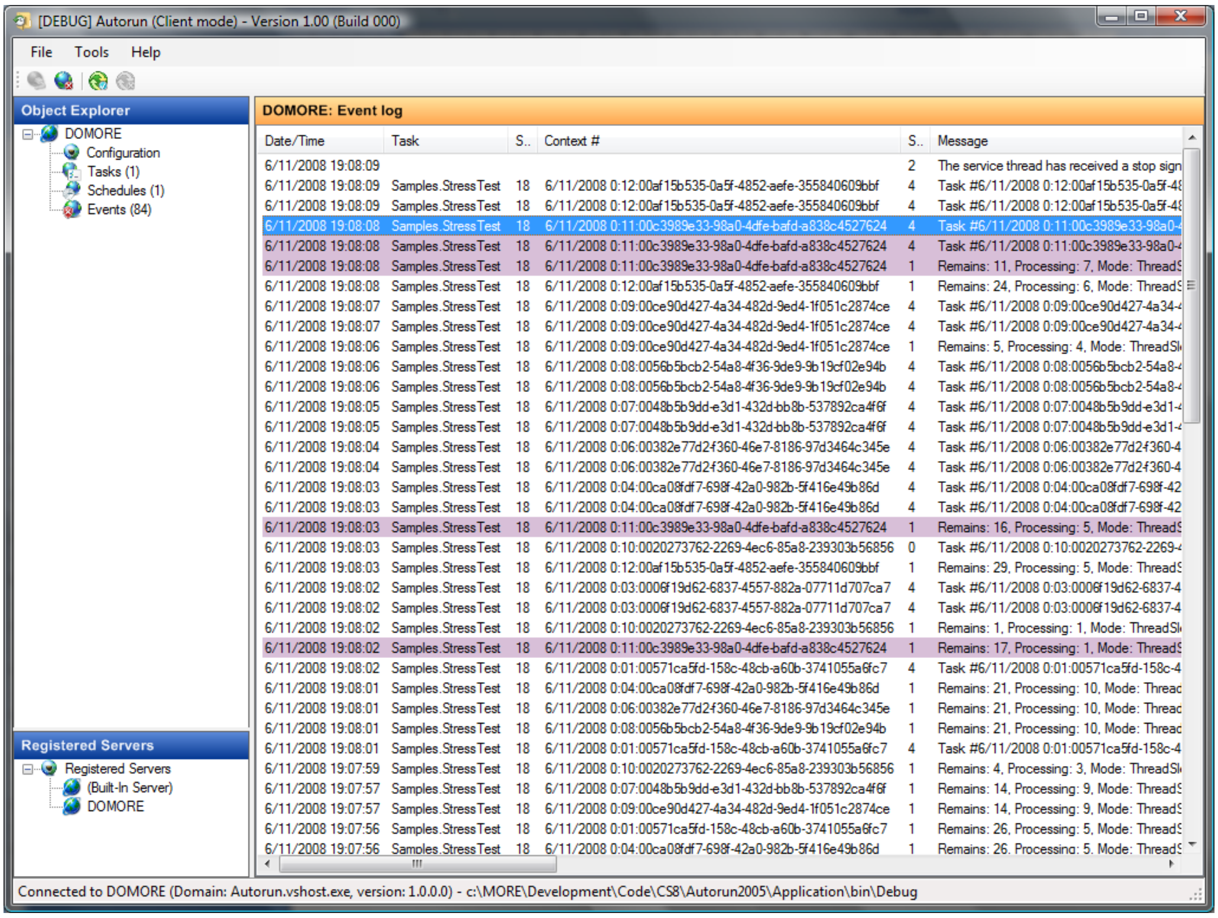
The log table is the table named “tblLog” and has the following structure:



Name	Req’d	Description
logId	Yes	Primary key
fktskId	No	Task # - When this field is null, it means the log entry is send by Autorun directly (i.e.: start/stop of the service)
fkschId	No	Shedule # - When no schedule, it means the task might have been started manually, or chained by another task.
logKey	Yes	This is a unique key representing the execution context. Filter on this key to retrieve all logs entries related to one specific execution
logDateTime	Yes	Date and time of the entry
logSeverity	Yes	Severity: verbose (debug), information, warning or error
logMessage	Yes	Message
logData	No	Additional data

The event log pane

You can also view the content of this table through the events pane in the client configuration tool.



DOMORE: Event log

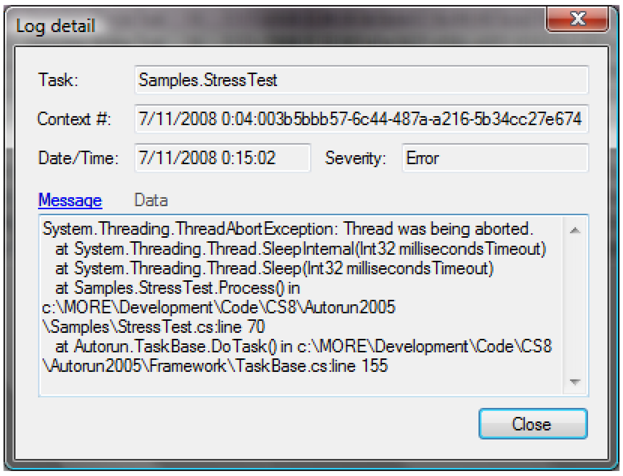
Date/Time	Task	S..	Context #	S..	Message
6/11/2008 19:08:09				2	The service thread has received a stop sign
6/11/2008 19:08:09	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	4	Task #6/11/2008 0:12:00af15b535-0a5f-48
6/11/2008 19:08:09	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	4	Task #6/11/2008 0:12:00af15b535-0a5f-48
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	4	Task #6/11/2008 0:11:00c3989e33-98a0-4
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	4	Task #6/11/2008 0:11:00c3989e33-98a0-4
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	1	Remains: 11, Processing: 7, Mode: ThreadS
6/11/2008 19:08:08	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	1	Remains: 24, Processing: 6, Mode: ThreadS
6/11/2008 19:08:07	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	4	Task #6/11/2008 0:09:00ce90d427-4a34-4
6/11/2008 19:08:07	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	4	Task #6/11/2008 0:09:00ce90d427-4a34-4
6/11/2008 19:08:06	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	1	Remains: 5, Processing: 4, Mode: ThreadSl
6/11/2008 19:08:06	Samples.StressTest	18	6/11/2008 0:08:0056b5bcb2-54a8-4f36-9de9-9b19cf02e94b	4	Task #6/11/2008 0:08:0056b5bcb2-54a8-4
6/11/2008 19:08:06	Samples.StressTest	18	6/11/2008 0:08:0056b5bcb2-54a8-4f36-9de9-9b19cf02e94b	4	Task #6/11/2008 0:08:0056b5bcb2-54a8-4
6/11/2008 19:08:05	Samples.StressTest	18	6/11/2008 0:07:0048b5b9dd-e3d1-432d-bb8b-537892ca4f6f	4	Task #6/11/2008 0:07:0048b5b9dd-e3d1-4
6/11/2008 19:08:05	Samples.StressTest	18	6/11/2008 0:07:0048b5b9dd-e3d1-432d-bb8b-537892ca4f6f	4	Task #6/11/2008 0:07:0048b5b9dd-e3d1-4
6/11/2008 19:08:04	Samples.StressTest	18	6/11/2008 0:06:00382e77d2f360-46e7-8186-97d3464c345e	4	Task #6/11/2008 0:06:00382e77d2f360-4
6/11/2008 19:08:04	Samples.StressTest	18	6/11/2008 0:06:00382e77d2f360-46e7-8186-97d3464c345e	4	Task #6/11/2008 0:06:00382e77d2f360-4
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	4	Task #6/11/2008 0:04:00ca08fdf7-698f-42
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	4	Task #6/11/2008 0:04:00ca08fdf7-698f-42
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	1	Remains: 16, Processing: 5, Mode: ThreadS
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:10:0020273762-2269-4ec6-85a8-239303b56856	0	Task #6/11/2008 0:10:0020273762-2269-4
6/11/2008 19:08:03	Samples.StressTest	18	6/11/2008 0:12:00af15b535-0a5f-4852-aefe-355840609bbf	1	Remains: 29, Processing: 5, Mode: ThreadS
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:03:0006f19d62-6837-4557-882a-07711d707ca7	4	Task #6/11/2008 0:03:0006f19d62-6837-4
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:03:0006f19d62-6837-4557-882a-07711d707ca7	4	Task #6/11/2008 0:03:0006f19d62-6837-4
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:10:0020273762-2269-4ec6-85a8-239303b56856	1	Remains: 1, Processing: 1, Mode: ThreadSl
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:11:00c3989e33-98a0-4dfe-bafd-a838c4527624	1	Remains: 17, Processing: 1, Mode: ThreadS
6/11/2008 19:08:02	Samples.StressTest	18	6/11/2008 0:01:00571ca5fd-158c-48cb-a60b-3741055a6fc7	4	Task #6/11/2008 0:01:00571ca5fd-158c-4
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	1	Remains: 21, Processing: 10, Mode: Thread
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:06:00382e77d2f360-46e7-8186-97d3464c345e	1	Remains: 21, Processing: 10, Mode: Thread
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:08:0056b5bcb2-54a8-4f36-9de9-9b19cf02e94b	1	Remains: 21, Processing: 10, Mode: Thread
6/11/2008 19:08:01	Samples.StressTest	18	6/11/2008 0:01:00571ca5fd-158c-48cb-a60b-3741055a6fc7	4	Task #6/11/2008 0:01:00571ca5fd-158c-4
6/11/2008 19:07:59	Samples.StressTest	18	6/11/2008 0:10:0020273762-2269-4ec6-85a8-239303b56856	1	Remains: 4, Processing: 3, Mode: ThreadSl
6/11/2008 19:07:57	Samples.StressTest	18	6/11/2008 0:07:0048b5b9dd-e3d1-432d-bb8b-537892ca4f6f	1	Remains: 14, Processing: 9, Mode: ThreadS
6/11/2008 19:07:57	Samples.StressTest	18	6/11/2008 0:09:00ce90d427-4a34-482d-9ed4-1f051c2874ce	1	Remains: 14, Processing: 9, Mode: ThreadS
6/11/2008 19:07:56	Samples.StressTest	18	6/11/2008 0:01:00571ca5fd-158c-48cb-a60b-3741055a6fc7	1	Remains: 26, Processing: 5, Mode: ThreadS
6/11/2008 19:07:56	Samples.StressTest	18	6/11/2008 0:04:00ca08fdf7-698f-42a0-982b-5f416e49b86d	1	Remains: 26, Processing: 5, Mode: ThreadS

Connected to DOMORE (Domain: Autorun.vshost.exe, version: 1.0.0.0) - c:\MORE\Development\Code\CS8\Autorun2005\Application\bin\Debug

The content of the table is not automatically updated. If you want to refresh its content, simply right click on it and select the "Refresh" menu entry on the contextual menu.

Event log detail

Some event log entries contain additional data that are not easy to read on the grid. Double click on the desired entry, or right click and select the menu item "View detail". The following screen appears:



Task: Samples.StressTest

Context #: 7/11/2008 0:04:003b5bbb57-6c44-487a-a216-5b34cc27e674

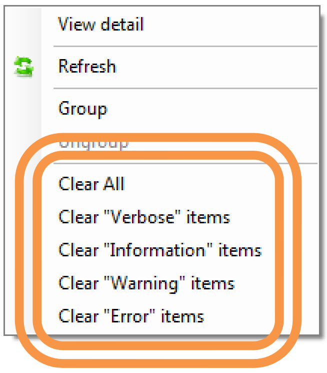
Date/Time: 7/11/2008 0:15:02 Severity: Error

```
System.Threading.ThreadAbortException: Thread was being aborted.
  at System.Threading.Thread.SleepInternal(Int32 millisecondsTimeout)
  at System.Threading.Thread.Sleep(Int32 millisecondsTimeout)
  at Samples.StressTest.Process() in
c:\MORE\Development\Code\CS8\Autorun2005
\Samples\StressTest.cs:line 70
  at Autorun.TaskBase.DoTask() in c:\MORE\Development\Code\CS8
\Autorun2005\Framework\TaskBase.cs:line 155
```

The message and data are displayed in a multiline format: you can also put the content of the textbox to the clipboard.

Deleting event log entries

You can clear the event log entirely or only entries related to a specific severity level. Right click on the grid and select between one of the following entries:

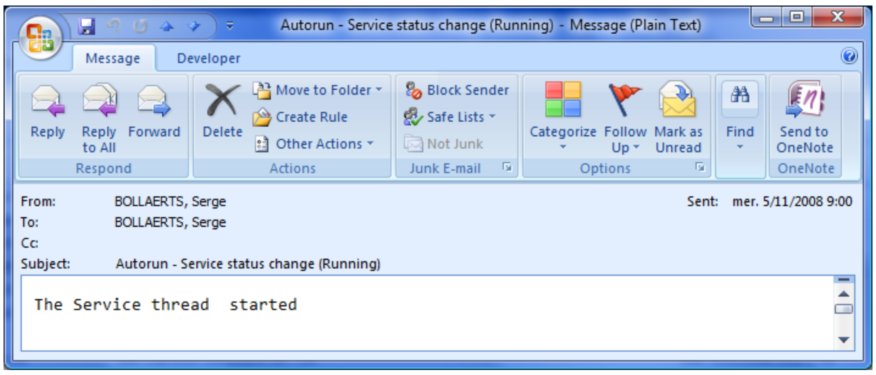


Service status changes

To be informed everytime the service has been started and stopped, configure the following setting:

Send Service Status	Should Autorun send an email everytime the server has been started and stopped?

The service status change report looks like:



From: BOLLAERTS, Serge
To: BOLLAERTS, Serge
Cc:
Subject: Autorun - Service status change (Running)
Sent: mer. 5/11/2008 9:00

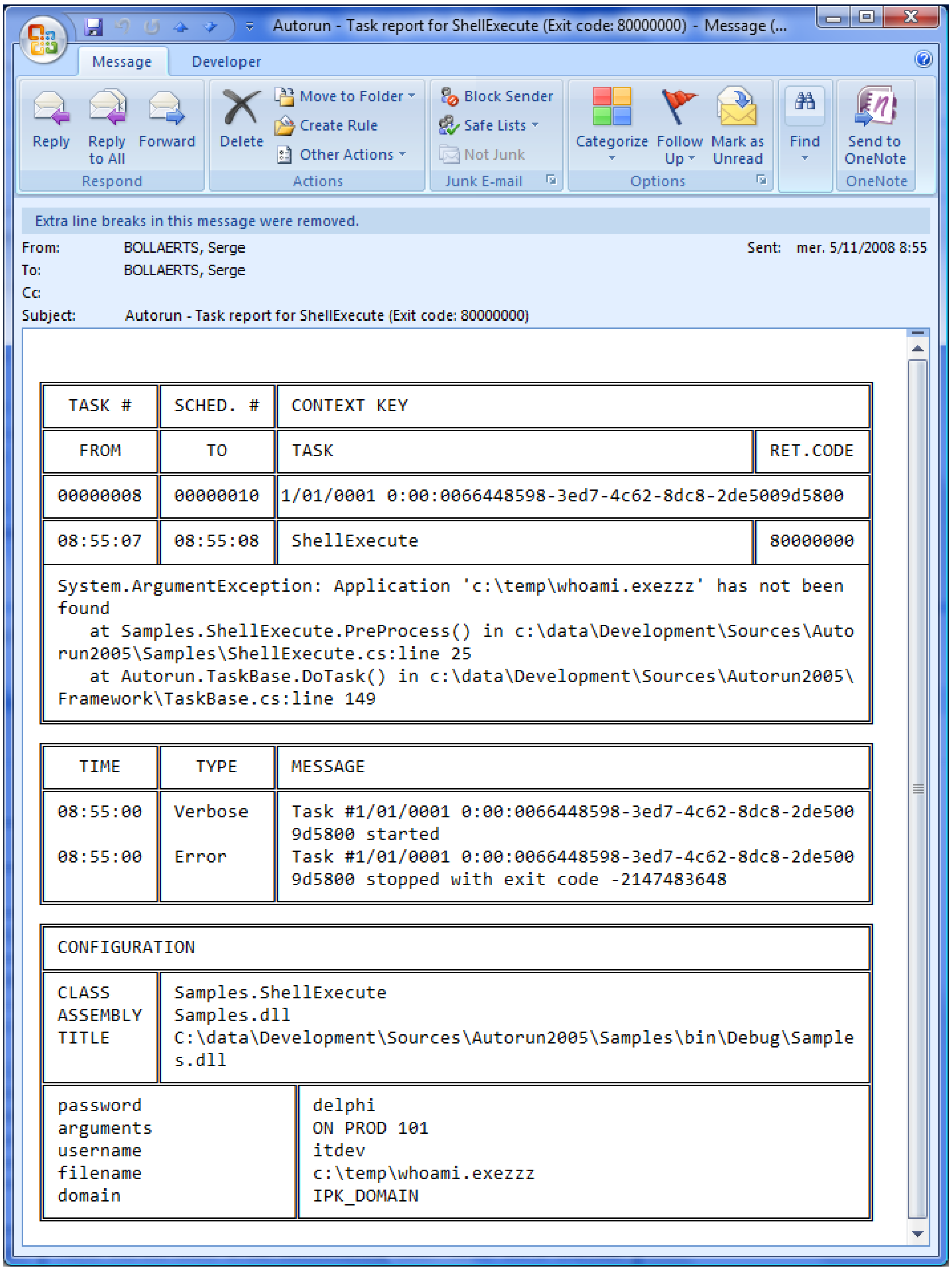
```
The Service thread  started
```

Task reports

You can be informed about task activity by setting the configuration key:

Send Task Reports	Define when task reports are sent: • Never: task reporting is disabled • OnErrors: when a task terminates with a return code <> 0, a mail is sent • Always: a task report is sent for each and every task execution

The task report looks like:



Extra line breaks in this message were removed.

From: BOLLAERTS, Serge Sent: mer. 5/11/2008 8:55
To: BOLLAERTS, Serge
Cc:
Subject: Autorun - Task report for ShellExecute (Exit code: 80000000)

TASK #	SCHED. #	CONTEXT KEY	
FROM	TO	TASK	RET.CODE
00000008	00000010	1/01/0001 0:00:0066448598-3ed7-4c62-8dc8-2de5009d5800	
08:55:07	08:55:08	ShellExecute	80000000

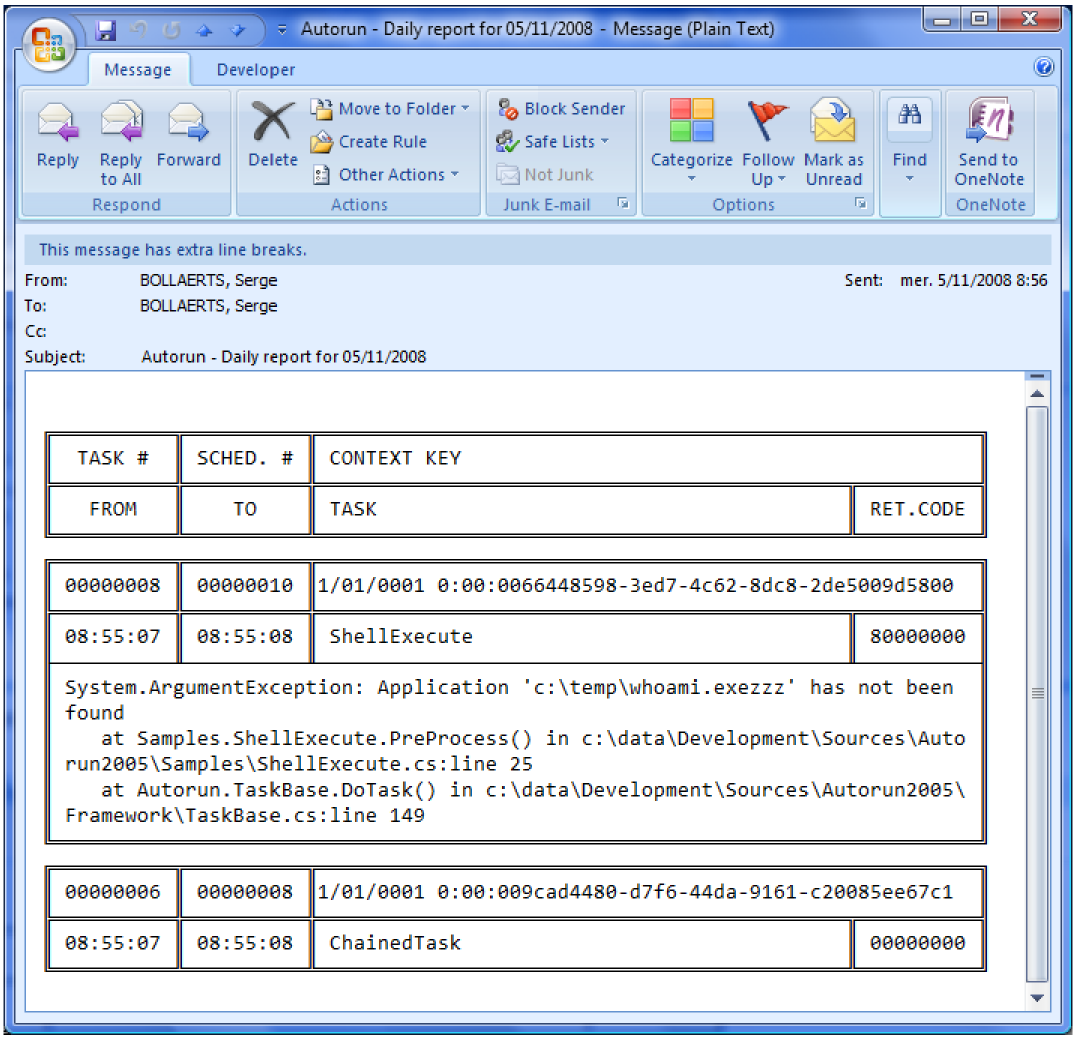
```
System.ArgumentException: Application 'c:\temp\whoami.exezzz' has not been
found
   at Samples.ShellExecute.PreProcess() in c:\data\Development\Sources\Auto
run2005\Samples\ShellExecute.cs:line 25
   at Autorun.TaskBase.DoTask() in c:\data\Development\Sources\Autorun2005\
Framework\TaskBase.cs:line 149
```

TIME	TYPE	MESSAGE
08:55:00	Verbose	Task #1/01/0001 0:00:0066448598-3ed7-4c62-8dc8-2de5009d5800 started
08:55:00	Error	Task #1/01/0001 0:00:0066448598-3ed7-4c62-8dc8-2de5009d5800 stopped with exit code -2147483648

CONFIGURATION	
CLASS	Samples.ShellExecute
ASSEMBLY	Samples.dll
TITLE	C:\data\Development\Sources\Autorun2005\Samples\bin\Debug\Samples.dll
password	delphi
arguments	ON PROD 101
username	itdev
filename	c:\temp\whoami.exezzz
domain	IPK_DOMAIN

Daily activity report

The daily report looks like:



This message has extra line breaks.

From: BOLLAERTS, Serge   Sent: mer. 5/11/2008 8:56
To: BOLLAERTS, Serge
Cc:
Subject: Autorun - Daily report for 05/11/2008

TASK #	SCHED. #	CONTEXT KEY	
FROM	TO	TASK	RET.CODE

00000008	00000010	1/01/0001 0:00:0066448598-3ed7-4c62-8dc8-2de5009d5800	
08:55:07	08:55:08	ShellExecute	80000000

```
System.ArgumentException: Application 'c:\temp\whoami.exezzz' has not been
found
   at Samples.ShellExecute.PreProcess() in c:\data\Development\Sources\Auto
run2005\Samples\ShellExecute.cs:line 25
   at Autorun.TaskBase.DoTask() in c:\data\Development\Sources\Autorun2005\
Framework\TaskBase.cs:line 149
```

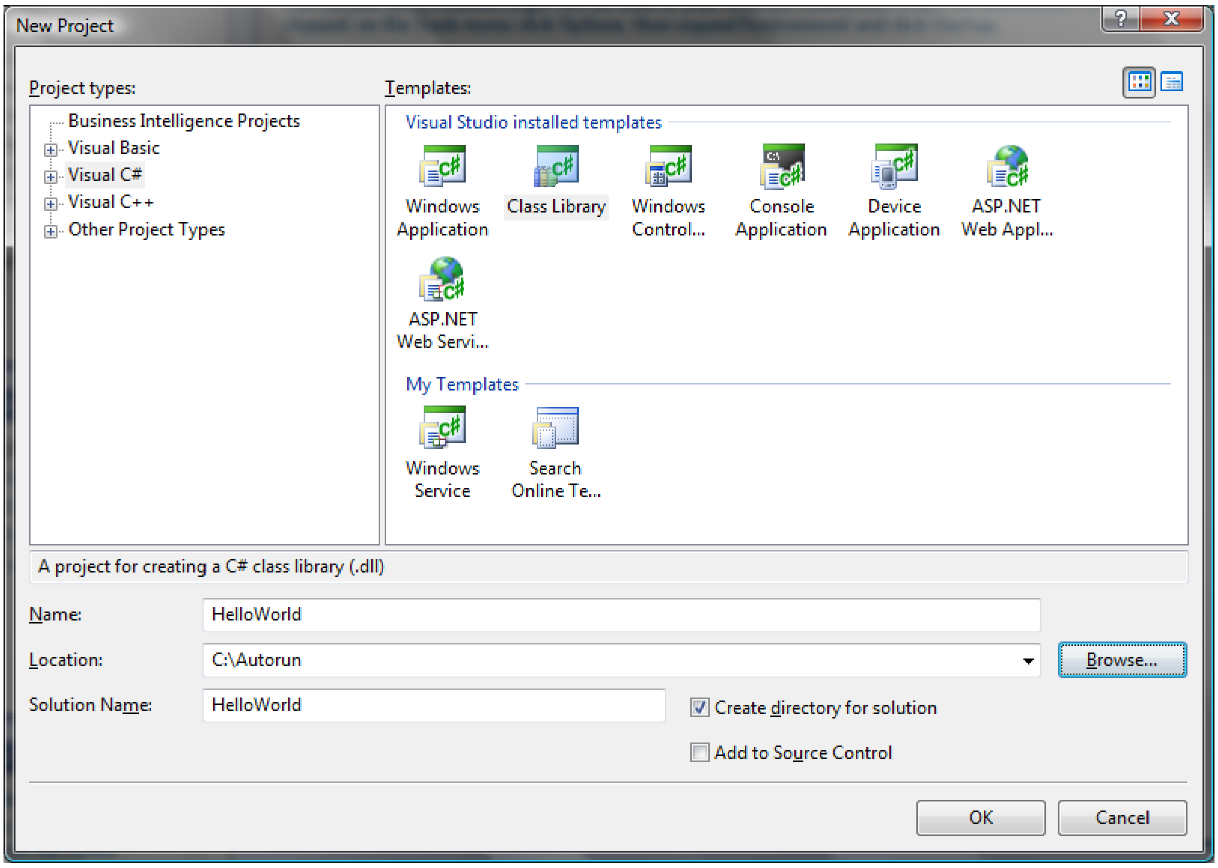
00000006	00000008	1/01/0001 0:00:009cad4480-d7f6-44da-9161-c20085ee67c1	
08:55:07	08:55:08	ChainedTask	00000000

Developer's guide

Quick start guide

In this tutorial you will learn how to create your first "Hello, world!" task. This tutorial assumes you have deployed Autorun into C:\Autorun.

1) Create a new project
   a. Start Visual Studio 2005 (or above)
   b. Create a new project using "Class Library" template, and name it HelloWorld
   c. Save it into C:\Autorun



A project for creating a C# class library (.dll)

Name: HelloWorld

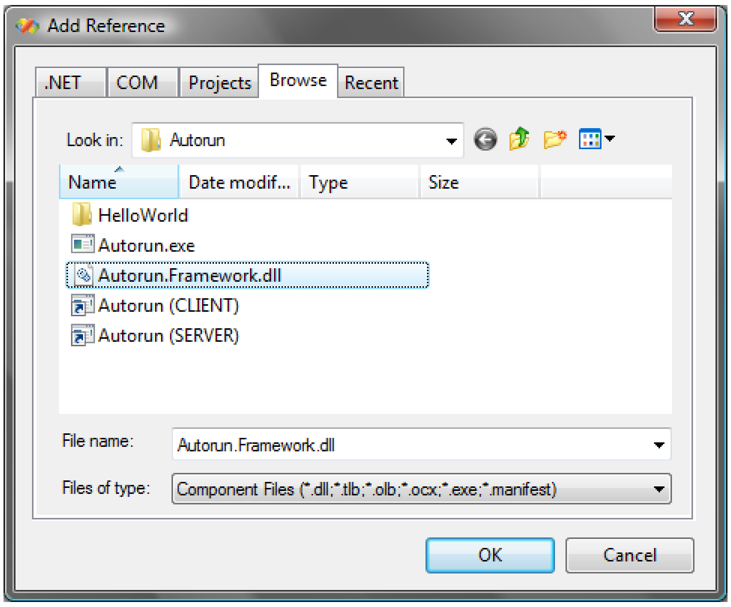
Location: C:\Autorun

Solution Name: HelloWorld

[✓] Create directory for solution

[ ] Add to Source Control

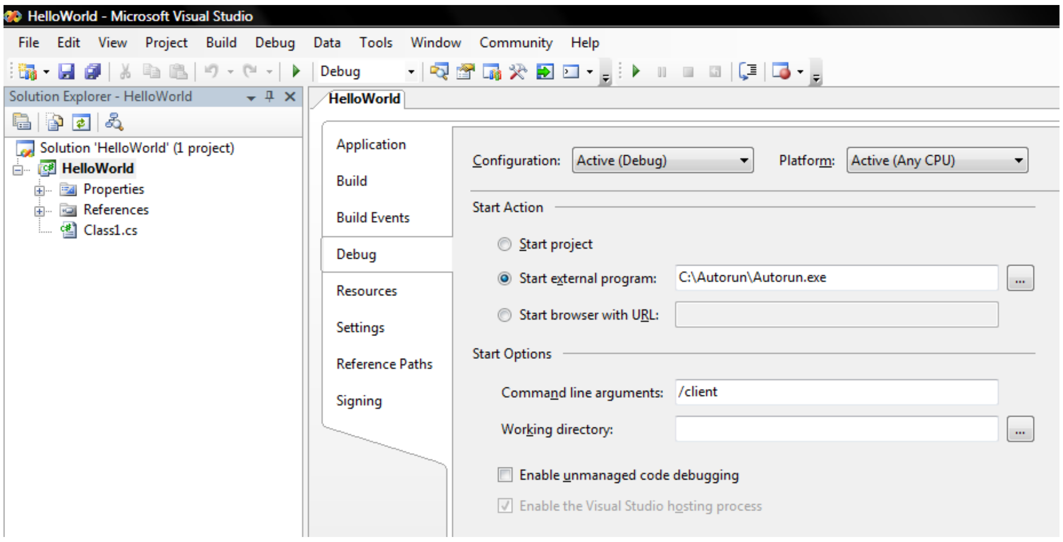
2) Add a reference to Autorun library
   a. Select the menu item "Add Reference…" under the menu entry "Project"
   b. Select the tab "Browse"
   c. Locate the library "Autorun.Framework.dll" located under C:\Autorun
   d. Click on "OK"



File name: Autorun.Framework.dll

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

3) Change the debug settings
    a. Select the menu item “HelloWorld Properties” under the menu entry “Project”
    b. Select the tab “Debug”
    c. Set the value of “Start external program” to “C:\Autorun\Autorun.exe”
    d. Set the value of “Command line arguments” to “/client”
    e. Save your settings
    f. Close the window


HelloWorld - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Debug
Solution Explorer - HelloWorld
Solution 'HelloWorld' (1 project)
HelloWorld
Properties
References
Class1.cs
HelloWorld
Application
Build
Build Events
Debug
Resources
Settings
Reference Paths
Signing
Start Action
Start project
Start browser with URL:
Start Options
Working directory:
Enable unmanaged code debugging
Enable the Visual Studio hosting process

Configuration: Active (Debug)   Platform: Active (Any CPU)

Start external program: C:\Autorun\Autorun.exe

Command line arguments: /client

4) Create your “HelloWorld” class
    a. Select the menu item “Add Class…” under the menu entry “Project”
    b. Name the class “HelloWorld”
    c. Click on “OK”
5) Add the references to used assembly

C#	`using Autorun;`
VB	`Import Autorun`

6) Indicate that the class derives from Task base

C#

```
using System;
using System.Collections.Generic;
using System.Text;
using Autorun;

namespace HelloWorld
{
        class HelloWorld : TaskBase
        {
                protected override void InitializeParameterList()
                {
                        throw new Exception("The method or operation is not implemented.");
                }

                protected override void PostProcess()
                {
                        throw new Exception("The method or operation is not implemented.");
                }

                protected override void PreProcess()
                {
                        throw new Exception("The method or operation is not implemented.");
                }

                protected override void Process()
                {
                        throw new Exception("The method or operation is not implemented.");
                }
        }
}
```

VB

```
Imports Autorun

Public Class HelloWorld
  Inherits TaskBase

  Protected Overrides Sub InitializeParameterList()

  End Sub

  Protected Overrides Sub PostProcess()

  End Sub

  Protected Overrides Sub PreProcess()

  End Sub

  Protected Overrides Sub Process()

  End Sub
End Class
```

7) Implement the parameter list initialization

C#

```
protected override void InitializeParameterList()
{
        Parameters.Add("YourName", "John Doe");
}
```

VB

```
Protected Overrides Sub InitializeParameterList()
  Parameters.Add("YourName", "John Doe")
End Sub
```

8) Implement the PreProcess function

C#

```
protected override void PreProcess()
{
        if ((Parameters["YourName"] == null) || (Parameters["YourName"].Length == 0))
                throw new ArgumentNullException("I cannot say hello to someone who doesn't
give his name!");
}
```

VB

```
Protected Overrides Sub PreProcess()
  If Parameters("YourName") Is Nothing OrElse Parameters("YourName").Length = 0 Then
    Throw New ArgumentNullException("I cannot say hello to someone who doesn't give his
name!")
  End If
End Sub
```

9) Implement the Process function

C#

```
protected override void Process()
{
        // === No clean up code needed =========================================
}
```

VB

```
Protected Overrides Sub Process()
  Notify(SeverityLevels.Verbose, String.Format("Hello, {0}!",
Parameters("YourName")))
End Sub
```

10) Implement the PostProcess function

C#

```
protected override void PostProcess()
{
        // === No clean up code needed =========================================
}
```

VB

```
Protected Overrides Sub PostProcess()
  ' === No cleanup code needed =============================================
End Sub
```

The complete class should now look like this:

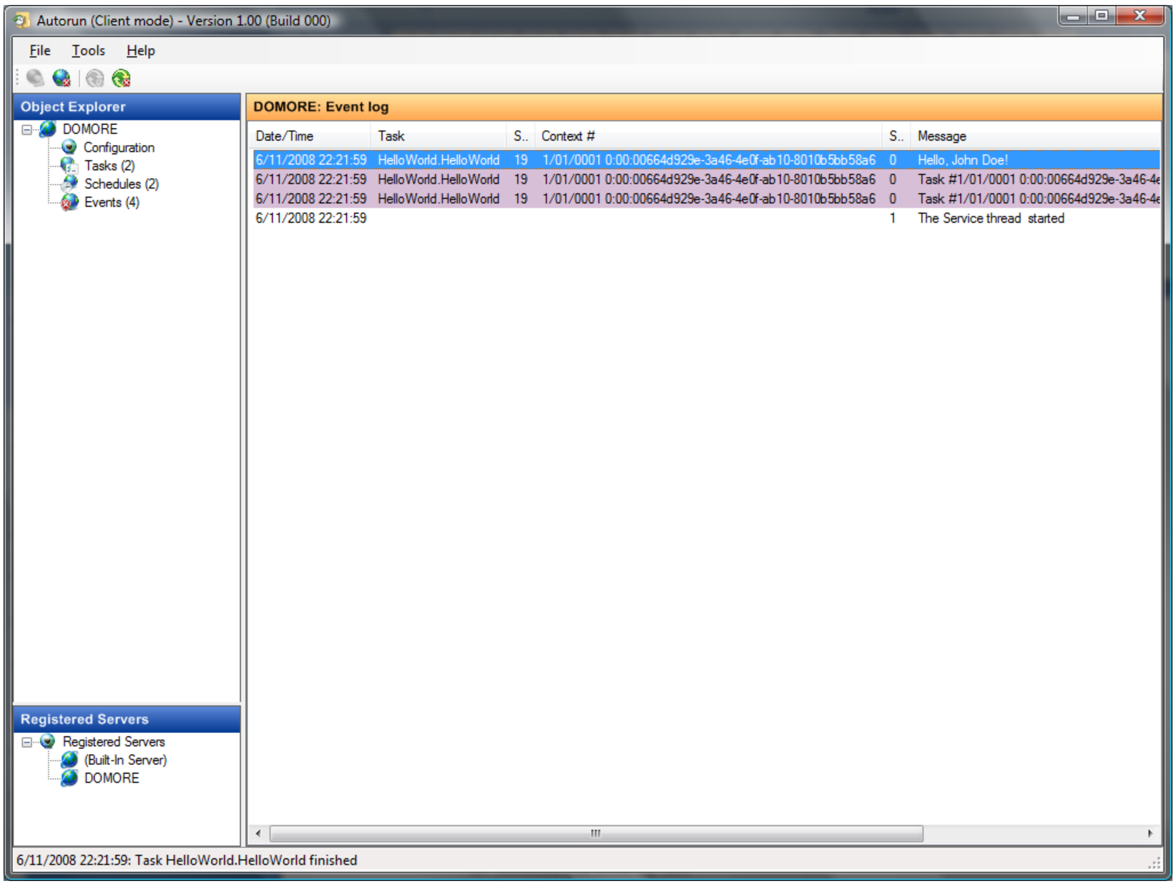
C#

```
using System;
using System.Collections.Generic;
using System.Text;
using Autorun;

namespace HelloWorld
{
        class HelloWorld : TaskBase
        {
                protected override void InitializeParameterList()
                {
                        Parameters.Add("YourName", "John Doe");
                }

                protected override void PostProcess()
                {
                        // === No clean up code needed
=========================================
                }

                protected override void PreProcess()
                {
                        if ((Parameters["YourName"] == null) ||
(Parameters["YourName"].Length == 0))
                                throw new ArgumentNullException("I cannot say hello to
someone who doesn't give his name!");
                }

                protected override void Process()
                {
                        Notify(SeverityLevels.Verbose, string.Format("Hello, {0}!",
Parameters["YourName"]));
                }
        }
}
```

VB

```vb
Imports Autorun

Public Class HelloWorld
  Inherits TaskBase

  Protected Overrides Sub InitializeParameterList()
    Parameters.Add("YourName", "John Doe")
  End Sub

  Protected Overrides Sub PostProcess()
    ' === No cleanup code needed =============================================
  End Sub

  Protected Overrides Sub PreProcess()
    If Parameters("YourName") Is Nothing OrElse Parameters("YourName").Length = 0 Then
      Throw New ArgumentNullException("I cannot say hello to someone who doesn't give his
name!")
    End If
  End Sub

  Protected Overrides Sub Process()
    Notify(SeverityLevels.Verbose, String.Format("Hello, {0}!", Parameters("YourName")))
  End Sub
End Class
```

11) Build your library and test it
    a. Run your project
    b. Connect to the built-in server
    c. Register your task
    d. Schedule it as a startup task
    e. Run the server
    f. Go to the event log



DOMORE: Event log

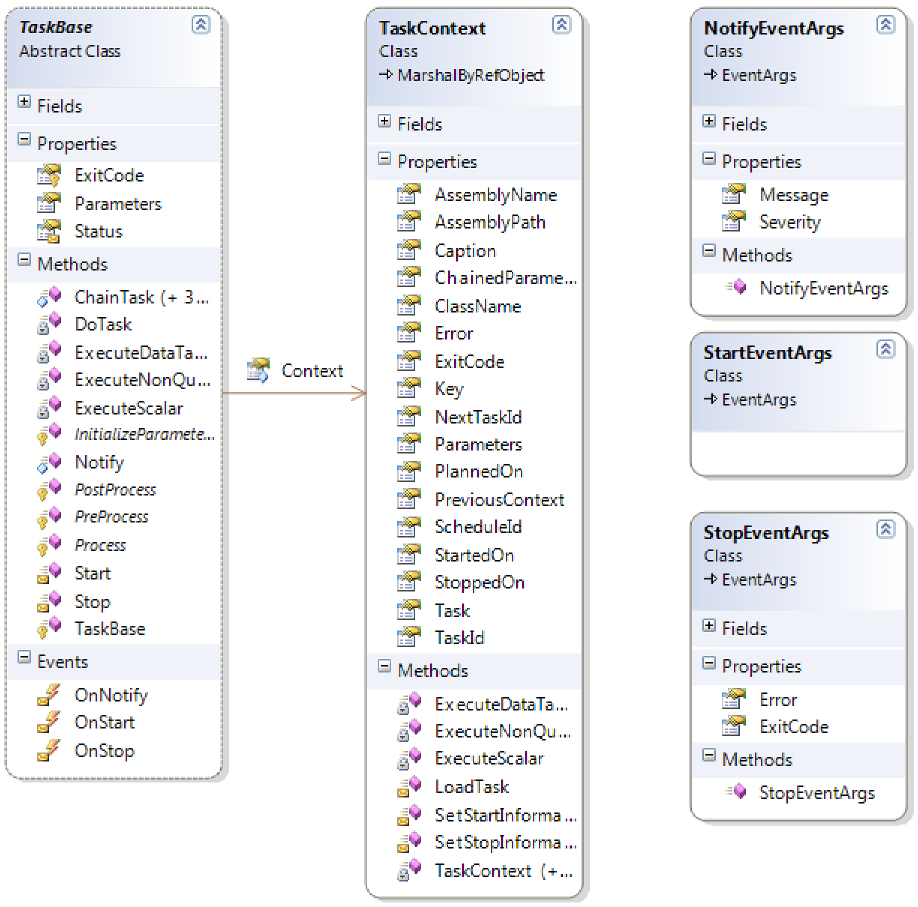
Date/Time	Task	S..	Context #	S..	Message
6/11/2008 22:21:59	HelloWorld.HelloWorld	19	1/01/0001 0:00:00664d929e-3a46-4e0f-ab10-8010b5bb58a6	0	Hello, John Doe!
6/11/2008 22:21:59	HelloWorld.HelloWorld	19	1/01/0001 0:00:00664d929e-3a46-4e0f-ab10-8010b5bb58a6	0	Task #1/01/0001 0:00:00664d929e-3a46-4e
6/11/2008 22:21:59	HelloWorld.HelloWorld	19	1/01/0001 0:00:00664d929e-3a46-4e0f-ab10-8010b5bb58a6	0	Task #1/01/0001 0:00:00664d929e-3a46-4e
6/11/2008 22:21:59				1	The Service thread started

6/11/2008 22:21:59: Task HelloWorld.HelloWorld finished

All you need to know about tasks

General presentation

Autorun manipulates your tasks by using the base class TaskBase. Every task must implement this class. The following figure describes the task base class and its relation with the TaskContext class.



Each and every task implementing this base class needs to implement at least 4 functions:

InitializeParameters	This function adds parameters keys and their default values to the collection Parameters
PreProcess	This is the first function called when a task is started: usually, you will put your initialization code here
Process	This is the main portion of your task
PostProcess	This is the last function called: usually, put your cleanup code here

The task can set a return code (the property ExitCode) and add a textual message in case of error (the property Error). Each task terminated with an exit code <> 0 is considered by Autorun as a task put in error.

Note
A later version will changed the interpretation of the exit code:
- >= 0: this is not considered as an error
- < 0: an error occured

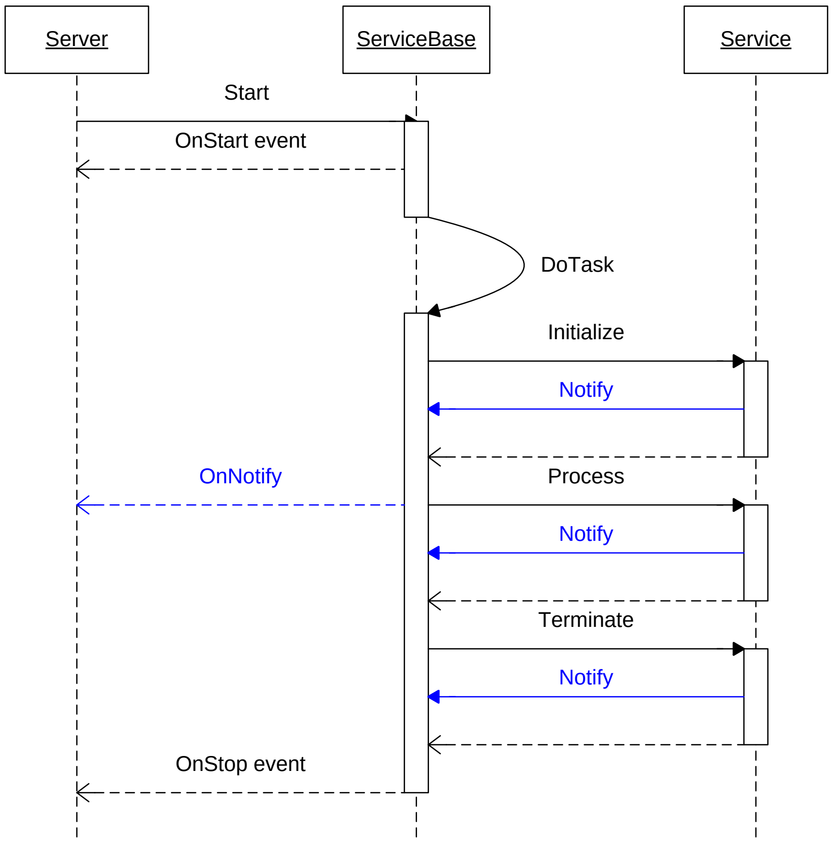
When exiting, a task can also ask Autorun to run another task: this is called “chaining tasks”. When chaining tasks, the task context is passed to the new task. You can also pass parameters to that chained task.

Task lifecycle

When the server executes a task, it does not know each and every task: they all must implement a common interface that will be provided through an abstract class.

In order to ensure multithreading, each task has its own thread. The server can start a task, or force stopping it in some exceptionnal circumstances (i.e.: stopping the server). The start signal will call sequentially 3 abstract functions:

- *Initialize:*
  - This will generally be the place where you verify the configuration parameters and perform other initializations.
- *Process:*
  - This will be the heart of the task.
- *Terminate:*
  - If any cleaning or data save must occurs, it might be done here.



Although it is not required to implement these functions, it could be a good practice separating pre-processing, processing and post-processing steps for readability.

Initialize, Process and Terminate do not return any code. However, it might be implemented in order to return an integer value. In any case, the server will interpret that code: if provided, it will save it in his log.

When an exception occurs, the task needs only to throw an exception. This exception will be caught by the server, and the task will be removed, preventing it from crashing other tasks or the server.

Sending notifications

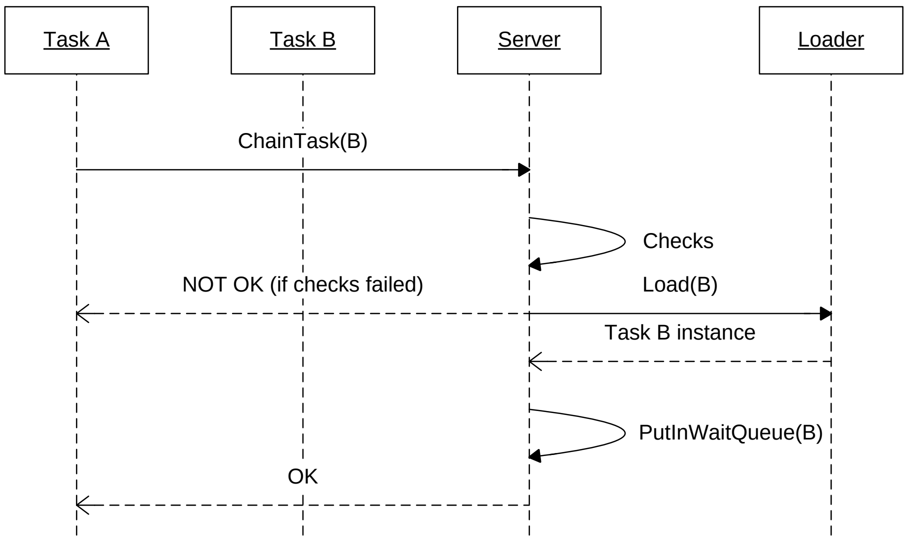
You can add messages in the event log by calling the function Notify:

- protected internal **void Notify**(**Autorun.SeverityLevels** *severity*, **string** *message*)

Chaining tasks

(At this time of writing the document, all specifications are not quite clear: this section will be subject to changes)

In some case, you will need to start another task to conditionally continue the job. In that case, the task will ask the server to chain a new task.



The above sequence diagram illustrates how starting a task should be implemented in the server. As you can notice, running task B is not done synchronously: the task B is put in a special waiting queue that has a higher priority over other planned task.

The following rules should be applied on the system:

- A task can only start one task
- The chained task must run only when the calling task terminates
- The chained task is started only when a free thread is available in the thread pool
- Only known tasks can be started dynamically: this means that the chained task must be at least declared in the server configuration, but it can be unscheduled.
- The chained task should receive in its execution context, information about the previous task(s)
  - It will be possible to pass data, or to get access to configuration of previous tasks

- The server should prevent cyclic task chaining: this is to ensure overall system stability.
  - This also means that a task A cannot ask to restart itself.

Chaining a task is done through the function ChainTask: there are 4 possible method calls:

- protected internal **void ChainTask**(**string** *assemblyPath*, **string** *assemblyName*, **string** *className*, **System.Collections.Specialized.StringDictionary** *chainedParameters*)
- protected internal **void ChainTask**(**string** *assemblyPath*, **string** *assemblyName*, **string** *className*)
- protected internal **void ChainTask**(**string** *assemblyName*, **string** *className*, **System.Collections.Specialized.StringDictionary** *chainedParameters*)
- protected internal **void ChainTask**(**string** *assemblyName*, **string** *className*)

The minimum parameters are the assembly name and the class name.

- The system searches for a registered task matching these criteria.
- When many tasks are found, the system returns an errors

A safer method is to specify a third parameter: the assembly path. With this parameter, there should be only one task matching these criteria.

The 4th parameter is the list of arguments you want to pass to the chained task.

Programming tips

Managing parameters

Because parameters are implemented as a StringDictionary, you can easily type an erroneous entry key without noticing it quickly.

For example:

```
Parameters.Add("YourName", "John Doe");
P = Parameters["YouName"];
```

➢ Declare constants representing your parameters keys

For example

```
private const string YOUR_NAME = "YourName";
Parameters.Add(YOUR_NAME, "John Doe");
P = Parameters[YOUR_NAME];
```

For people who are not used to, this could look like a waste of time, but I can tell you it really worth it because this kind of error does not result to a fatal error…

The Pre-Process function

As you know, parameters are stored as string data types: it means you can encounter parsing errors. Use the function PreProcess to validate all input parameters before starting the main process. You can also initialize strong-typed members that your will use instead the parameters collection.

For example

```
using System;
using System.Collections.Generic;
using System.Text;
using Autorun;
using System.Windows.Forms;

namespace HelloWorld
{
	class HelloWorld : TaskBase
	{
		private const string YOUR_NAME = "YourName";

		private string _name;

		protected override void InitializeParameterList()
		{
			Parameters.Add(YOUR_NAME, "John Doe");
		}

		protected override void PostProcess()
		{
			// === No clean up code needed =========================================
		}

		protected override void PreProcess()
		{
			if ((Parameters[YOUR_NAME] == null) || (Parameters[YOUR_NAME].Length ==
0))
				throw new ArgumentNullException("I cannot say hello to someone
who doesn't give his name!");
			else
				_name = Parameters[YOUR_NAME];
		}

		protected override void Process()
		{
			Notify(SeverityLevels.Verbose, string.Format("Hello, {0}!", _name));
			MessageBox.Show(string.Format("Hello, {0}!", _name));
		}
	}
}
```

Quickly debug your task

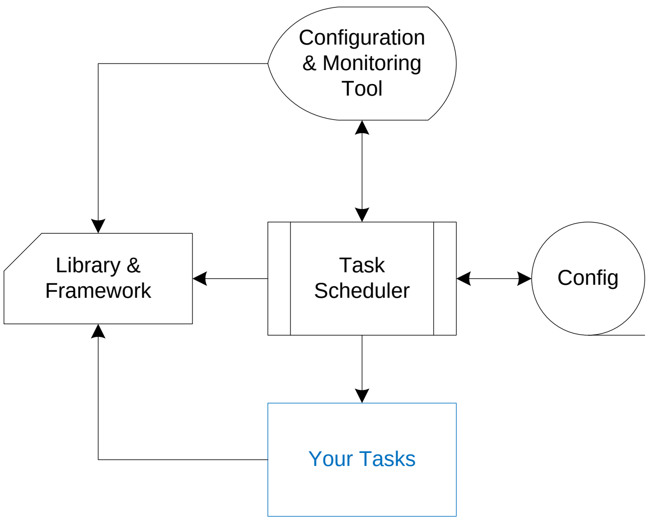
When you develop a new task you do not require the complete Autorun environment. Instead you could create a small application that instantiates your task and run it. You can now do this by invoking the method **Test()** of your task.

The standard Way to use it should be

```
SomeTask item;
StringDictionary parameters;

item = new SomeTask();

item.Parameters["ConnectionString"] = Properties.Settings.Default.ConnectionString;
item.Test();
```

Solution architecture

The goal of the solution was to provide the following target components.



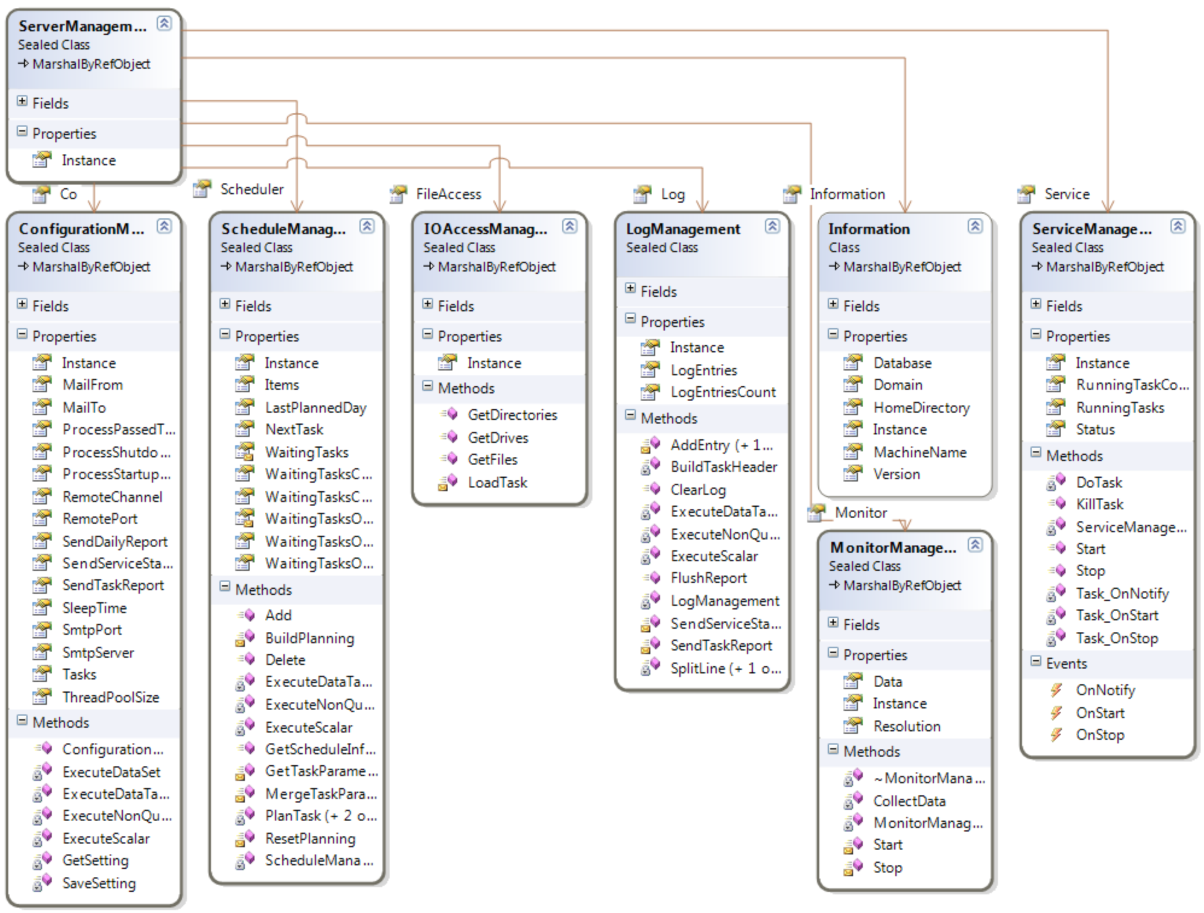
To achieve this goal, only two projects were needed:

- **Application:**
  - Represents the executable, whatever the starting mode
- **Library:**
  - Includes all classes and interfaces needed to create tasks or build your own executable.

Each class forming the "Autorun" server is developed to be remotely accessed by a client application. When connecting the server, the entry point is the class "Server Management". You can you this class to access other objects of the server. The following table presents the classes of Autorun:

ServerManagement	This is the main entry point that is instantiated when connected remotely
ServiceManagement	This is the main service
ConfigurationManagement	Is responsible for managing system configuration
ScheduleManagement	This object builds daily planning
IOAccessManagement	This object centralizes access methods related to loading task objects

LogManagement	This object provides logging functionalities
Information	Provides general information about the service
MonitorManagement	Collects data used to monitor the activity of the service



System requirements

Development environment

The following environment has been used to develop the solution.

Operating system	Windows XP or above
Development tools	Visual studio 2005
Database components	SQL Server 2005 or above
System components	Framework.NET 2.0
	SMTP Sever (i.e: IIS/SMTP service) if you plan to receive emails

Runtime environment

Operating system	Windows XP or above
Database components	SQL Server 2005 or above
System components	Framework.NET 2.0 or above
	SMTP Sever (i.e: IIS/SMTP service) if you plan to receive emails