Introduction

Procrun is a set of applications that allow Windows users to wrap (mostly) Java applications (e.g. Tomcat) as a Windows service.

The service can be set to automatically start when the machine boots and will continue to run with no user logged onto the machine.

Procrun monitor application

Prunmgr is a GUI application for monitoring and configuring procrun services.

Each command line directive is in the form of **//XX[//ServiceName]**

If the `//ServiceName` parameter is omitted, then the service name is assumed to be the name of the file.
The Prunsrv application behaves in the same way, so to allow both applications to reside in the same directory, the Prunmgr application will remove a trailing **w** (lower-case w) from the name.
For example if the Prunmgr application is renamed as `TestService.exe` - or as `TestServicew.exe` - then the default service name is `TestService`.

The available command line options are:

//ES	Edit service configuration	This is the default operation. It is called if the no option is provided. Starts the GUI application which allows the service configuration to be modified, started and stopped.
//MS	Monitor service	Starts the GUI application and minimizes it to the system tray.
//MR	Monitor & run service	Starts the GUI application and minimizes it to the system tray. Start the service if it is not currently running.
//MQ	Monitor Quit	Stop any running monitor for the service.

Procrun service application

Prunsrv is a service application for running applications as services. It can convert any application (not just Java applications) to run as a service.

Command line arguments

Each command line directive is in the form of **//XX[//ServiceName]**.

If the `//ServiceName` parameter is omitted, then the service name is assumed to be the name of the file.
For example if the application is renamed as `TestService.exe`, then the default service name is `TestService`.

The available command line options are:

//TS	Run the service as a console application	This is the default operation. It is called if the no option is provided.
//RS	Run the service	Called only from ServiceManager
//ES	Start (execute) the service	
//SS	Stop the service	
//US	Update service parameters	
//IS	Install service	
//DS	Delete service	Stops the service first if it is currently running
//PP[//seconds]	Pause	Default is 60 seconds
//VS	Version	Print version and exit (since version 1.0.3)
//?	Help	Print usage and exit (since version 1.0.3)

Starting with version **1.0.8** a more traditional command line can be used in the form: **command [ServiceName]**.

run	Run the service as a console application	This is the default operation. It is called if the no option is provided and has the same effect as calling **//TS**.
service	Run the service	Called only from ServiceManager
start	Start the service	Synonym for **//ES**
stop	Stop the service	Synonym for **//SS**
update	Update service parameters	Synonym for **//US**
install	Install service	Synonym for **//IS**
delete	Delete service	Stops the service first if it is currently running
pause [seconds]	Pause	Default is 60 seconds
version	Version	Print version and exit
help	Help	Print usage and exit

Command line parameters

Each command parameter is prefixed with **--** (or **++**, see below).
If an environment variable exists with the same name as a command line parameter but prefixed with PR_ it will **override** the equivalent command line parameter.
For example:

```
set PR_CLASSPATH=xx.jar
```

is equivalent to providing

```
--Classpath=xx.jar
```

as a command line parameter.

If a parameter is repeated, then normally the last value takes precedence. However some parameters can take multiple values - for example StartParams and JvmOptions. If these parameters are prefixed with **++**, then the value will be appended to the existing value. For example:

```
--Startup=manual --Startup=auto --JvmOptions=-Done=1 ++JvmOptions=-Dtwo=2
```

will result in the following values being used:

```
Startup:
auto

JvmOptions:
-Done=1
-Dtwo=2
```

Only multi-valued parameters support this; they are indicated in the table below by **++**.

If **++** is used for a parameter that does not support multiple values, then it is treated the same as **--**. No error is reported. Configuration is overwritten in case **--** is used. For example:

```
--JvmOptions=-Dthree=3 ++JvmOptions=-Dfour=4
```

will always overwrite the JvmOptions. The resulting configuration will be:

```
Startup:
auto

JvmOptions:
-Dthree=3
-Dfour=4
```

However if on **++** is used the values will be appended. For example calling the following after the first example

```
++JvmOptions=-Dthree=3 ++JvmOptions=-Dfour=2
```

will result in the following values being used:

```
Startup:
auto

JvmOptions:
-Done=1
-Dtwo=2
-Dthree=3
-Dfour=4
```

In case you intermix the **++** and **--** options, the last **--** parameter will cause option reset. For example:

```
--Startup=manual --Startup=auto --JvmOptions=-Done=1 ++JvmOptions=-Dtwo=2 --JvmOptions=-Dthree=3 ++JvmOptions=-Dfour=2
```

will result in the following values being used:

```
Startup:
auto

JvmOptions:
```

```
-Dthree=3
-Dfour=4
```

When updating a service (//US or update command), using **--** will replace any existing parameter with the new setting. For multi-valued parameters, using the **++** option qualifier will add the new value(s) to any existing value(s).

Parameter Name	Default	Description
--Description		Service name description (maximum 1024 characters)
--DisplayName	ServiceName	Service display name
--Install	procrun.exe //RS//ServiceName	Install image
--Startup	manual	Service startup mode can be either **auto** or **manual**
--Type		Service type can be **interactive** to allow the service to interact with the desktop. Use this option only with Local system accounts.
++DependsOn		List of services that this service depends on. Dependent services are separated using either **#** or **;** characters
++Environment		List of environment variables that will be provided to the service in the form **key=value**. They are separated using either **#** or **;** characters. If you need to embed either # or ; character within a value put them inside single quotes.
--User		User account used for running executable. It is used only for StartMode **Java** or **exe** and enables running applications as a service under an account without the LogonAsService privilege.
--Password		Password for user account set by --User parameter
--ServiceUser		Specifies the name of the account under which the service should run. Use an account name in the form *DomainName**UserName*. The service process will be logged on as this user. if the account belongs to the built-in domain, you can specify *.\UserName*
--ServicePassword		Password for user account set by --ServiceUser parameter
--LibraryPath		Directory added to the search path used to locate the DLLs for the JVM. This directory is added both in front of the **PATH** environment variable and as a parameter to the **SetDLLDirectory** function.
--JavaHome	JAVA_HOME	Set a different JAVA_HOME than defined by JAVA_HOME environment variable
--Jvm	auto	Use either **auto** (i.e. find the JVM from the Windows registry) or specify the full path to the **jvm.dll**. You can use environment variable expansion here.
++JvmOptions	-Xrs	List of options in the form of **-D** or **-X** that will be passed to the JVM. The options are separated using either **#** or **;** characters. if you need to embed either # or ; character put them inside single quotes. (Not used in **exe** mode.)
--Classpath		Set the Java classpath. (Not used in **exe** mode.)
--JvmMs		Initial memory pool size in MB. (Not used in **exe** mode.)
--JvmMx		Maximum memory pool size in MB. (Not used in **exe** mode.)
--JvmSs		Thread stack size in KB. (Not used in **exe** mode.)
--StartMode		One of **jvm**, **Java** or **exe**. The modes are: • jvm - start Java in-process. Depends on jvm.dll, see **--Jvm**. • Java - same as exe, but automatically uses the default Java executable, i.e. %JAVA_HOME%\bin\java.exe. Make sure JAVA_HOME is set correctly, or use --JavaHome to provide the correct location. If neither is set, procrun will try to find the default JDK (not JRE) from the Windows registry. • exe - run the image as a separate process
--StartImage		Executable that will be run. Only applies to **exe** mode.
--StartPath		Working path for the start image executable.
--StartClass	Main	Class that contains the startup method. Applies to the **jvm** and **Java** modes. (Not used in **exe** mode.)
--StartMethod	main	Name of method to be called when service is started. It must be `static void` and have argument `(String args[])`. Only applies to **jvm** mode - in **Java** mode, the **main** method is always used.

		Note: in `jvm` mode, the start method should not return until the stop method has been called.
++StartParams		List of parameters that will be passed to either StartImage or StartClass. Parameters are separated using either **#** or **;** character.
--StopMode		One of **jvm**, **Java** or **exe**. See **--StartMode** for further details.
--StopImage		Executable that will be run on Stop service signal. Only applies to **exe** mode.
--StopPath		Working path for the stop image executable. Does not apply to **jvm** mode.
--StopClass	Main	Class that will be used on Stop service signal. Applies to the **jvm** and **Java** modes.
--StopMethod	main	Name of method to be called when service is stopped. It must be `static void` and have argument `(String args[])`. Only applies to **jvm** mode. In **Java** mode, the **main** method is always used.
++StopParams		List of parameters that will be passed to either StopImage or StopClass. Parameters are separated using either **#** or **;** character.
--StopTimeout	No Timeout	Defines the timeout in seconds that procrun waits for service to exit gracefully.
--LogPath	%SystemRoot%\System32\LogFiles\Apache	Defines the path for logging. Creates the directory if necessary.
--LogPrefix	commons-daemon	Defines the service log filename prefix. The log file is created in the LogPath directory with `.YEAR-MONTH-DAY.log` suffix
--LogLevel	Info	Defines the logging level and can be either **Error**, **Info**, **Warn** or **Debug**. (Case insensitive).
--LogJniMessages	0	Set this non-zero (e.g. 1) to capture JVM jni debug messages in the procrun log file. Is not needed if stdout/stderr redirection is being used. Only applies to **jvm** mode.
--StdOutput		Redirected stdout filename. If named **auto** file is created inside **LogPath** with the name **service-stdout.YEAR-MONTH-DAY.log**.
--StdError		Redirected stderr filename. If named **auto** file is created in the **LogPath** directory with the name **service-stderr.YEAR-MONTH-DAY.log**.
--PidFile		Defines the file name for storing the running process id. Actual file is created in the **LogPath** directory

Installing services

To install the service, you need to use the **//IS** parameter.

Install the service named 'TestService'

```
prunsrv //IS//TestService --DisplayName="Test Service" \
        --Install=prunsrv.exe --Jvm=auto --StartMode=jvm --StopMode=jvm \
        --StartClass=org.apache.SomeStartClass --StartParams=arg1;arg2;arg3 \
        --StopClass=org.apache.SomeStopClass --StopParams=arg1#arg2
```

Updating services

To update the service parameters, you need to use the **//US** parameter.

Update the service named 'TestService'

```
prunsrv //US//TestService --Description="Some Dummy Test Service" \
        --Startup=auto --Classpath=%CLASSPATH%;test.jar
```

Removing services

To remove the service, you need to use the **//DS** parameter. If the service is running it will be stopped and then deleted.

Remove the service named 'TestService'

```
prunsrv //DS//TestService
```

Debugging services

To run the service in console mode, you need to use the **//TS** parameter. The service shutdown can be initiated by pressing **CTRL+C** or **CTRL+BREAK**. If you rename the prunsrv.exe to testservice.exe then you can just execute the testservice.exe and this command mode will be executed by default.

Run the service named 'TestService' in console mode

```
prunsrv //TS//TestService [additional arguments]
```

Using Procrun in jvm mode

To interface with the Procrun service application (prunsrv) using the **jvm** mode, you need to create a class with the appropriate method(s). For example:

```
class MyClass;
// N.B. error handling not shown
static void main(String [] args){
    String mode = args[0];
    if ("start".equals(mode){
        // process service start function
    }
    etc.
}
```

This should be configured as follows:

```
--Classpath MyClass.jar
--StartMode jvm --StartClass MyClass --StartParams start
--StopMode  jvm --StopClass  MyClass --StopParams  stop
```

The above example uses a single 'main' method, and uses a string parameter to specify whether the service function is start or stop.

Alternatively, you can use different method names for the service start and stop functions:

```
class MyClass;
// N.B. error handling not shown
static void start(String [] args){
        // process service start function
    }
static void stop(String [] args){
        // process service stop function
    }
}
```

This should be configured as follows:

```
--Classpath MyClass.jar
--StartMode jvm --StartClass MyClass --StartMethod start
--StopMode  jvm --StopClass  MyClass --StopMethod  stop
```

Note that the method handling service start should create and start a separate thread to carry out the processing, and then return. The start and stop methods are called from different threads.

Using Procrun in Java or exe mode

When using the **Java** or **exe** modes, the Procrun service application (prunsrv) launches the target application in a separate process. The "stop" application needs to communicate somehow with the "start" application to tell it to stop. For example, using RPC.

Windows Registry Usage

The basic Service definitions are maintained under the registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<ServiceName>
```

Additional parameters are stored in the registry at:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\ProcRun 2.0\<ServiceName>\Parameters
```

On 64-bit Windows procrun always uses 32-bit registry view for storing the configuration. This means that parameters will be stored inside:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Apache Software Foundation\ProcRun 2.0\<ServiceName>
```