Running the Server As a Service

The Servoy Application Server comes bundles with an Java Service Wrapper component, to facilitate the automatic launch of the Servoy Application Server when the server on which the Servoy Application Server is installed is started or rebooted.

⚠ The Java Service Wrapper does not start the required databases as the databases are not part of the Servoy Application Server, could be running on a completely different machine and different brands of databases have different mechanisms to starting. For more information on running the bundled PostgreSQL as a service, see the PostgreSQL Primer tutorial.

When running the Servoy Application Server and the database on the same Windows machine, both as a Windows Service, then a dependency can be configured between the different Windows services. See Configuring Windows Service Dependencies for more info.

In This Chapter

- Supported Platforms
- Installation
- Additional Configuration
  - Configuring the Windows Service to Automatically Start on Reboot of the Machine
  - Configuring Windows Service Dependencies
  - Memory Setting
  - Running Multiple Servoy Application Servers As a Service on a Single Machine
    - Windows
    - OSX/*nix
    - All Platforms
  - Additional Configurations
    - Perform Custom Action on OutOfMemory Exceptions
  - More Info on the Java Service Wrapper Component
- Running the Windows Service from a Different User

Supported Platforms

The Java Service Wrapper used by Servoy supports virtually all platforms on which a Java Virtual Machine is available (also required to run the Servoy Application Server).

The Java service Wrapper component will create a Windows Service on the Windows platform and on other platforms it utilizes Unix Daemons.

Installation

The Java Service Wrapper is by default installed by the Servoy All-In-One installer when installing a Servoy Application Server (see Running the All-In-One Installer for more info on the All-In-One installer).

During installation, an extra `service` directory is created in the `../application_server/` directory. This directory contains, among others, the configuration file for the service, called `wrapper.conf`.

If the **Setup Service** option is checked in the installer, the Service will actually be setup on the machine. By default this option is unchecked, as in most cases the service configuration will require changes.

The Service component installation comes with two files to install and uninstall the Service: `install_service.`[bat/sh] and `uninstall_service.`[bat/sh] which are also found in the `service` directory.

Additional Configuration

The configuration of the Java Service Wrapper is done through the file `wrapper.conf` located in `../application_server/service.`

⚠ When changing the value of any any `wrapper.ntservice.*` property on Windows, the Service needs to be uninstalled first and reinstalled after the changes have been made.

Configuring the Windows Service to Automatically Start on Reboot of the Machine

By default the Windows Service is configured to be started automatically. This can be changed to start manually by setting the `wrapper.ntservice.starttype` property to `DEMAND_START`.

⚠ Note that the service starts successfully only if the database is already running at that point.

Configuring Windows Service Dependencies

The Java Service Wrapper does NOT start the database, since in production environments the database could be installed on a different machine or is maintained by a DBA.

If the database is running on the same machine as a Windows Service as well, it is possible to configure a dependency to it in the Java Service Wrapper configuration. The dependency can be set by adding the name of the Windows Service for the database on which the Servoy Application Service depends as a `wrapper.ntservice.dependency.x` property. The 'x' at the end of the property name needs to be a sequential number, starting with 1, for each dependency.

Memory Setting

When the Servoy Application server is started as a service through the Java Service Wrapper, the assignment of memory goes through the configuration of the wrapper in `wrapper.conf`, instead of through `servoy_server.`[bat/sh].

Below the mapping for the three relevant memory settings and their equivalent in the `wrapper.conf`:

- Initial Java Heap Size > `wrapper.java.initmemory=...`
- Maximum Java Heap Size > `wrapper.java.maxmemory=...`
- Maximum Java Perm Size > `wrapper.java.additional.4=-XX:MaxPermSize=...`

For more information on the the meaning of these settings and their suggested values, see Memory Management.

Running Multiple Servoy Application Servers As a Service on a Single Machine

When running multiple Servoy Application Servers as a Service on a single machine, some configuration adjustments are required to not let the different services influence each other.

Windows

The following property requires a unique value:

- `wrapper.ntservice.name` (default `ServoyService`)

On Windows the `wrapper.ntservice.displayname` and `wrapper.ntservice.description` could be modified to further distinguish the different Windows Services from each other. Note that these two settings are just for display purposes in the Services overview in Windows and do not affect the actual operation of the services.

OSX/*nix

On OSX and *nix, the name used for the deamon is configured in ServoyServer:

```
# Application
APP_NAME="ServoyServer"
APP_LONG_NAME="Servoy Application Server"
```

The `APP_NAME` has to be unique for each Service.

All Platforms

The `wrapper.port` value does not have to be unique, because the Java Service Wrapper automatically looks for a free port if the specified port is already taken.

Additional Configurations

The Java Service Wrapper has extensive configuration options. The most fundamental settings have been discussed in this chapter already. In this paragraph some more advanced settings are detailed.

Perform Custom Action on OutOfMemory Exceptions

Java processes need to be explicitly told how much memory they are allowed to use. In case the process requires more memory than it's allowed to use based on the configuration, Java will throw OutOfMemory exceptions. When this happens, the state of the process becomes unreliable, as some vital business logic might not have been executed. Java only has limited options to take action when OutOfMemory exceptions occur.

The Java Service Wrapper however supports a monitoring feature that is capable of taking action in case that OutOfMemory exceptions occur. The following snippet of configuration will perform a thread dump and then a restart of the Servoy Application Server.

```
wrapper.filter.trigger.1=Exception in thread "*" java.lang.OutOfMemoryError
wrapper.filter.action.1=NONE

wrapper.filter.trigger.2=Exception in thread "*" java.lang.OutOfMemoryError
wrapper.filter.allow_wildcards.2=TRUE
wrapper.filter.action.2=DUMP,RESTART
wrapper.filter.message.2=The JVM has run out of memory.
```

More Info on the Java Service Wrapper Component

The Java Service Wrapper is a bundled third party component also called Java Service Wrapper Standard by Tanuki Software. For in depth view of the subject have a look on their documentation.

The component is licensed by Servoy to be bundled and used in combination with the Servoy Application Server only. It's not allowed to use the component outside the context of the Servoy Application Server.

Running the Windows Service from a Different User

A non-admin user doesn't have permission to start and stop the Servoy Application Server service. In order to have access to do that, they have to be granted permission by the administrator. This section provides information on how to grant permission to a different user to access the Servoy service.

The following steps need to be followed in order to give access to the service:

1. Login to an administrator account and run a command prompt console. Make sure to run the console as administrator.
2. Enter the following command in the console:

   ```
   C:\>sc sdshow servoyservice
   ```

   The output will look something like this:

   ```
   D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;CCLCSWLOCRRC;;;IU)(A;;CCLCSWLOCRRC;;;
   SU)S:(AU;FA;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)
   ```

   Copy and paste it into a text editor.
3. Get the SID (Security IDentifier) of the user accound that needs permission to access the service.
   To obtain it, do the following:
   a) go to **Start** > **regedit** and locate this registry path:
   `LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList`
   All the user accounts are displayed under this node, and the key name of each account is the SID.
   b) Click on each key and look on the right panel at the **ProfileImagePath** to identify the user name of the account.
   c) When the desired user is identified, copy its key name, that is the SID, and paste it in the text editor on a new line.
4. In the text editor:
   a) copy the `(A;;CCLCSWRPWPDTLOCRRC;;;SY)` part of the command output, and paste it right before the `S:(AU;FA;...` part
   b) change that part to look like this: `(A;;RPWPCR;;;[SID])` - where [SID] has to be replaced by the actual user account SID
   c) surround the whole resulted line with double quotes, and copy it
5. In the command prompt, enter the `sc sdset` command and paste in the copied line. The resulted command will look like this:

   ```
   sc sdset servoyservice "D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;
   CCLCSWLOCRRC;;;IU)(A;;CCLCSWLOCRRC;;;SU)(A;;RPWPCR;;;S-1-5-21-30086777-1921087056-1718559101-1003)S:(AU;FA;
   CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)"
   ```

6. Run the command. On success, the following output should appear:

   ```
   [SC] SetServiceObjectSecurity SUCCESS
   ```

After completing these steps, the non-admin user for which these settings have been done will be able to access the Application Server service.